

N69-30271

NASA CR-86155

NAS 12-665
CR-86155

012

First Interim Report

NASA _____

RESEARCH IN THE EFFECTIVE IMPLEMENTATION
OF

GUIDANCE COMPUTERS WITH LARGE SCALE ARRAYS

BY J. J. Pariser
F. D. Erwin
J. F. McKevitt
J. A. Burke
C. P. Disparte
C. H. Schardin, Editor

CASE FILE
COPY

October 1968
(Revised May 1969)

Distribution of this report is provided in the interest of information exchange and should not be construed as endorsement by NASA of the material presented. Responsibility for the contents reside with the organization that prepared it.

FR 69-11-621

Prepared under Contract No. NAS 12-665 by
Hughes Aircraft Company
Fullerton, California

Electronics Research Center

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Dr. Harold E. Maurer
Technical Monitor
Electronics Research Center
Cambridge, Massachusetts

Requests for copies of this report should be
referred to:

NASA Scientific and Technical Information Facility
P. O. Box 33, College Park, Maryland 20740

First Interim Report

NASA

RESEARCH IN THE EFFECTIVE IMPLEMENTATION
OF
GUIDANCE COMPUTERS WITH LARGE SCALE ARRAYS

BY J. J. Pariser
F. D. Erwin
J. F. McKeivitt
J. A. Burke
C. P. Disparte
C. H. Schardin, Editor

October 1969
(Revised May 1969)

FR 69-11-621

Prepared under Contract No. NAS 12-665 by
Hughes Aircraft Company
Fullerton, California

Electronics Research Center

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

TABLE OF CONTENTS

Sections	Page No.
Topics	
Sub-Topics	
Summary	1
Introduction	3
Description of Building Blocks	5
G1 Character	6
L1 Character	7
L2 Character	8
L3 Character	9
M1 Character	10
M2 Character	11
MM Character	12
P1 Character	12
P2 Character	13
P3 Character	14
Description of Microprogram Repertoire	15
Micro-Memory Word	15
Instruction Fields	15
Source Field	15
Operator Sub-field	17
Destination Sub-field	18
Transfer Bits	19
Preliminary Implementation of the MCB Computer	21
General Design Considerations	21
Control Unit	21
Arithmetic Unit	25
Memory Unit	25
Input-Output Unit	25
Configuration Assignment Unit	25
Control Unit Design	26
Arithmetic Unit Design	30
Memory Unit Design	33

TABLE OF CONTENTS (cont'd)

Sections	Page No.
Topics	
Sub-Topics	
Input-Output Unit Design	36
Configuration Unit Design	39
MCB Execution Rates	42
Evaluation of Preliminary Implementation of MCB	45
Considerations	45
Feasibility of Logic Design Techniques	45
Types of Functional Characters	46
Speed of Operation	46
Power, Size and Weight	46
Reliability	46
Conclusions	47
Special Purpose Computers	48
Digital Differential Analyzers	48
Serial-Parallel Conversion	52
Spaceborne Computer Characteristics	53
Weighted Design Considerations for Aerospace Computers	57
Reliability	57
Producibility	62
Number of LSI Types	62
LSI Package Size	62
Number of Terminals	63
Testability	63

TABLE OF CONTENTS (cont'd)

Sections	Page No.
Topics	
Sub-Topics	
Operational Environment	63
Special Circuits	63
System Integration	63
Packaging Method	63
References	64
Appendices	

LIST OF ILLUSTRATIONS

Figure No.	Title	Page No.
1	Typical Character Configuration	5
2	G1 Character Block Diagram	6
3	L1 Character Block Diagram	7
4	L2 Character Block Diagram	8
5	L3 Character -- I/O Interface Block Diagram	9
6	M1 Character Block Diagram	10
7	M2 Character Block Diagram	11
8	MM Character Block Diagram	12
9	P1 Character Block Diagram	12
10	P2 Character Block Diagram	13
11	P3 Character Block Diagram	14
12	Micromemory Half-Word ($b_0 - b_{47}$)	15
13	Instruction Field	15
14	MCB - Modular Computer Breadboard Block Diagram	22
15	MCB Control Unit - CU - Block Diagram	27
16	Control Unit Microprogram Flow Chart - General	28
17	Control Unit Microprogram Flow Chart - Detailed Example	29
18	MCB Arithmetic Unit - AU Block Diagram	31
19	Arithmetic Unit Microprogram Flow Chart	32
20	MCB Memory Unit - MU Block Diagram	34
21	Memory Unit Microprogram Flow Chart	35
22	MCB Input-Output Unit - I/O Block Diagram	37
23	Input-Output Microprogram Flow Chart	38
24	MSB - Configuration Assignment Unit - CAU Block Diagram	40
25	Configuration Assignment Unit Flow Chart	41
26	Integrator Unit Block Diagram	49
27	DDA Block Diagram	50
28	Sample Flow Chart for DDA	51

LIST OF TABLES

TABLE	TITLE	PAGE
I	Source Mnemonics	16
II	List of Control Unit Subroutines	24
III	MCB Execution Times	43
IV	MCB Execution Times After Modification	44
V	Spaceborne Computer Characteristics	54
VI	Mean-Life Comparison	59

(First Interim Report)

RESEARCH IN THE EFFECTIVE IMPLEMENTATION
OF
GUIDANCE COMPUTERS WITH LARGE SCALE ARRAYS

By J. J. Pariser
F. D. Erwin
J. F. McKevitt
J. A. Burke
C. P. Disparte

Hughes Aircraft Company
Fullerton, California

SUMMARY

This report is the first of two resulting from a study of an effective method of implementing guidance computers. The approach used applies functional logic characters to the selected system design of the NASA Modular Computer (NMC) as represented by a modular computer breadboard (MCB). The characters are specified as the building blocks and a repertoire of microprogram instructions is defined. The character set totals ten unique building blocks. There are three logic characters, one register character, three micromemory-related characters and three miscellaneous characters (scratch pad memory, real-time clock and switch). A micromemory word contains instruction and constant fields. Based on available information of the MCB Computer, a design of each of the five units was completed using the functional characters. The design of each unit in general includes selecting proper characters and microprogramming each micromemory to provide proper logical interaction of characters within a unit and between units to furnish all specified MCB machine instructions and operations. An evaluation of the implementation of the MCB using functional characters is given. The preliminary implementation results in favorable operating time and a reasonable hardware count.

The report includes sections on special-purpose computers, spaceborne computer characteristics, and weighted design characteristics for aerospace computers. It is demonstrated in the section on special-purpose computers that the same functional character set can readily implement a special-purpose computer. The other sections encompass reports prepared to give a current overview of the related subjects on aerospace computers. These sections point-up the wisdom of the functional character set approach to implementing the upcoming generation of digital systems.

The character set has been chosen from the logical design point of view for the moment holding in abeyance some of the design guidelines, thus the gates per character are in the vicinity of 300; the gates per chip need not be as large however. A character could contain several chips in some mechanical and electrical arrangements. The subpartitioning of characters into chips with about 100 gates per chip is the next phase of this study.

Since the characters represent a logical complement, they can be implemented with almost any type of circuitry that meets the speed-power considerations. The study is proceeding utilizing T²L circuits as stipulated in the work statement. However Hughes strongly recommends that Ion-implanted MOS circuits receive parallel consideration as the implementation vehicle. The decision does not need to be made until the commencement of a prototype design of a computer. Ion implantation is required if MOS is to be used to meet the speed and power objectives. The reliability and availability risks of the ion implanted MOS circuits are reasonable and manageable.

Acknowledgements:

The participation and consultation of NASA ERC members - Dr. E. Bersoff, Messrs. J. Bowe, M. Liematainen, W. Lurie, and C. Watt, in particular, are gratefully acknowledged. We also express our appreciation to Drs. H. Maurer and F. Tung for their encouragement and guidance.

INTRODUCTION

Logical Design of digital equipments is based primarily on Boolean logic with some sprinkling of switching theory. However, by and large, the design is an individual matter, lacking standardization, with each designer retracing the steps of his colleague. The methods utilized in programming such as use of macros, higher order languages, etc., are relatively unknown in logical design. This study endeavors to introduce standardization to the task of logic design in a form characteristic of a higher order language. To this end, a set of functional characters has been designed and successfully tested for sufficiency of implementing various digital equipments. "Implementation" referred herein and throughout the report implies in the logical sense and not in the circuit sense. The difference between these lies in the realization of physical circuits which represent the logic and at the same time accommodate the various electrical, thermal, and mechanical constraints. A character in this report refers to a logical entity which can be subpartitioned to suit the physical implementation requirements. The work associated with taking the logical specifications to a physical realization involves many tradeoffs and interactions of various disciplines. This work will be a subject of further efforts and studies.

A significant portion of this report is devoted to demonstrating the application of a minimal set of general-purpose functional characters (logic arrays) to the implementation of an existing logic design of a general-purpose computer. The reader is therefore introduced first to the set of logical building blocks. He is then presented with a repertoire of micro-instructions and microprogram-word structure suitable for controlling the desired logic functions of the specified machine instructions. Following the presentation of this basic background material, the detailed preliminary implementation of the MCB Computer is given. In general, each unit of the MCB is implemented with the functional characters as shown in a block diagram form. A flow diagram presents the microprogram operations for general unit operation and, in some cases, for specific detailed examples. Some general observations and conclusions related to the functional character implementation are presented.

The flexibility of the functional characters is demonstrated by performing a first-cut design of a digital differential analyzer (DDA) using the same techniques applied to the implementation of the MCB.

The last two sections of this report are in fact, separate reports prepared as part of the overall study on aerospace computers. These reports cover the topics of aerospace computer characteristics and weighted design characteristics.

The reader is advised to view this report and effort with its purpose in mind. Namely to demonstrate the feasibility (not practicability) of implementing various digital systems using predesigned functionally-organized logical groupings - functional character set. The fact that the MCB implementation resulted in comparable speeds with the custom-designed logic is indeed an unexpected benefit, since no optimization of the architecture for use of functional characters has been attempted. The purpose of the implementation was simply to demonstrate conclusively that purely on a logical basis equipments can be designed with pre-interconnected logic. Our intent was to completely ignore the circuit aspects (speed, fan-in, fan-out, etc.). Now that the feasibility of designing digital equipments using macro-logical elements (functional characters) has been established, we will proceed according to our work plan with the refinements of the character set and procedures in order to obtain the most efficient design in terms of production risk, reliability, performance, etc.

DESCRIPTION OF BUILDING BLOCKS

This section describes a set of building blocks for implementing the logic of digital systems. The building blocks are defined as characters.*

The functional character set is a group of logic arrays forming a self-sufficient family of blocks which reduce computer design to a determination of character types and number. For the design of the MCB, 10 character types are required as listed below. They are described in greater detail in the following paragraphs:

G1	Register storage	P1	Scratch pad memory
L1	General logic	P2	Up/Down counter
L2	Arithmetic logic	P3	Switch
L3	Input/Output		
M1	Micromemory counter		
M2	Micro-instruction Register		
MM	Micro-array		

In some cases the characters resemble the commercially available complex functions (MSI), the difference being in systematic design approach, versatility, and completeness of the set to form systems. In contrast, complex function characteristics tend to be derived from statistics of usage in present computer designs.

Characters of the same letter are logically grouped into a common unit as illustrated in the diagram below: (Figure 1)

92577-3

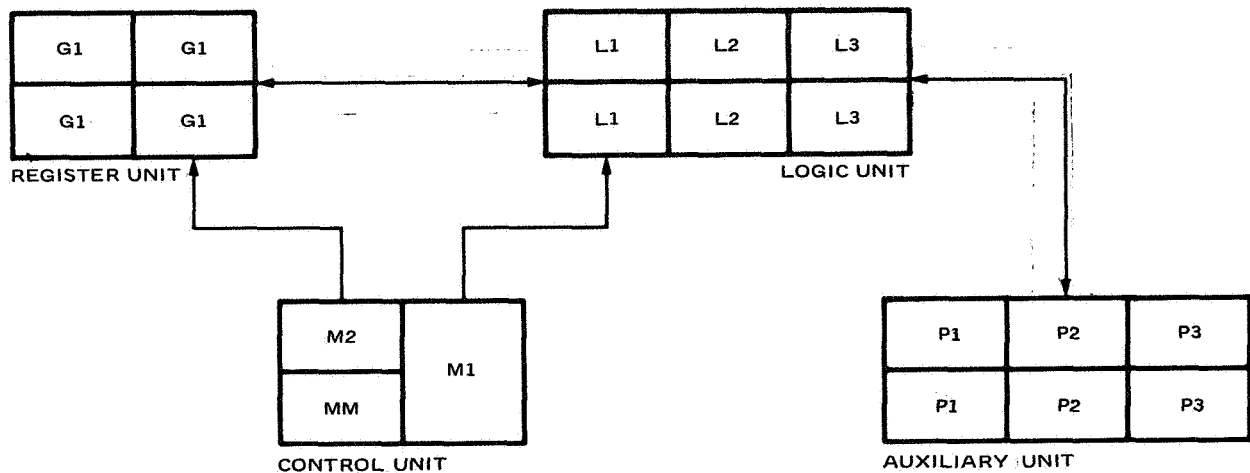


Figure 1. Typical Functional Character Configuration

*The words "character" and "card" are used interchangeably. (It is not to be inferred that the logic content of a functional character will necessarily represent the contents of a circuit card.)

G1 Character

The G1 character provides the bulk of storage for operands of the micro-program. Each character contains 4 registers of 8 bits each accompanied by reading and writing selector gates.

The input buss carries information to the character. The data from the buss is distributed within the characters under micro-program control, with the command decoding being part of the character. Similarly, the output of the character is presented on the output buss under microprogram control.

The conceptual structure of the G1 character is shown in Figure 2.

92577-1

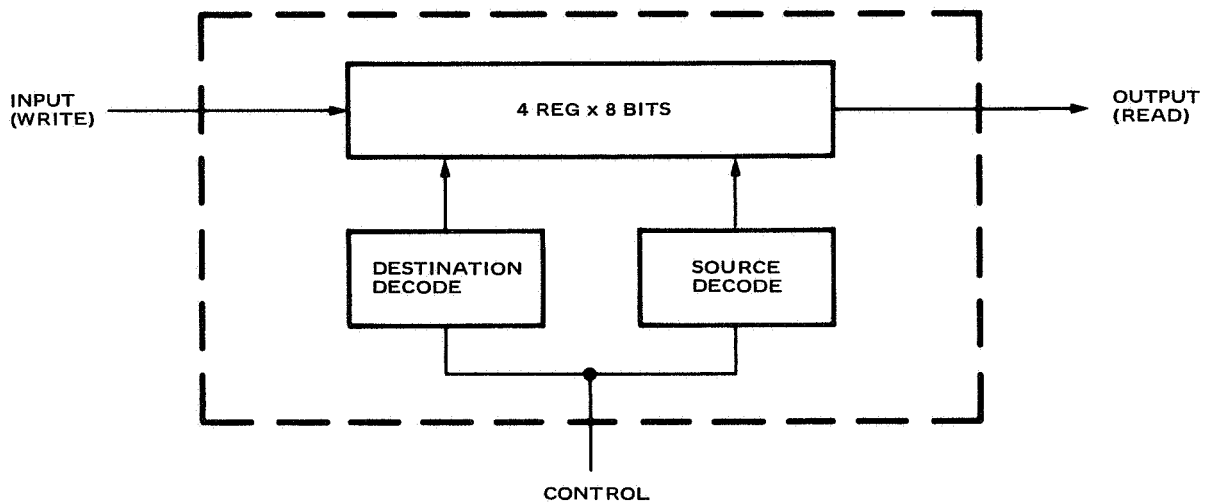


Figure 2. G1 Character Block Diagram

L1 Character

The L1 character provides the basic logic functions for use by the micro-program. These functions consist of the rotates, shifts (logical), transfer, complement, "AND", and incrementation. Also associated with the L1 character is the decoding logic for these logic operations. Incrementation and some other functions are accomplished with the use of a logic register. The L1 character is 8 bits wide and contains the following logic:

1. Decoding logic
2. Rotate, shift, and complement logic
3. Incrementer
4. L register
5. Gating to output bus

In Figure 3 is shown a block diagram for the L1 character.

92577-2

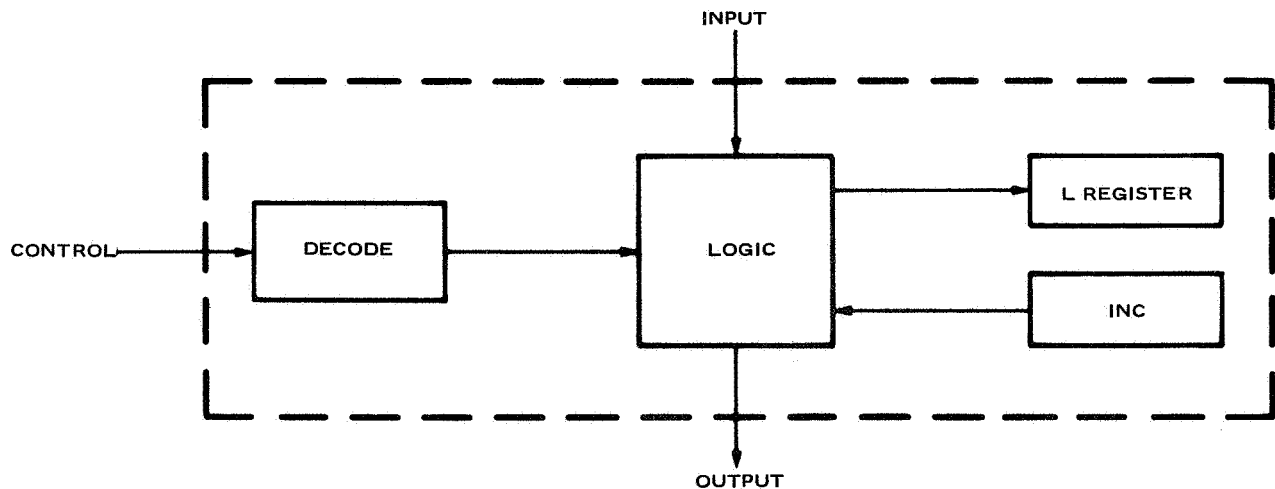


Figure 3. L1 Character — Block Diagrams

Several L1 characters may be connected together to form logic operations on words longer than one byte.

Information to the L1 character is presented at the input. Thereafter the information operated upon by the logic function is selected from the current micro-memory word and decoded in this character (left side of Figure). The resultant is available at the output where it leaves the character or is optionally stored in the L register (where it would thus be available at the next micro-instruction time for internal use.

L2 Character

The L2 character provides the major arithmetic functions for use by the micro-program. Addition is performed with carry look-ahead byte parallel. Control signals may condition the adder to alternately provide either of two special results (a) a mod 2 addition instead of full addition or (b) an input carry to the lowest order bit for full addition. All arithmetic is in 2's complement form. The L2 character consists of two holding registers for the operands of the adder, the adder itself, decoding and error logic, and bussing gates. Figure 4 diagrams function-wise the L2 character.

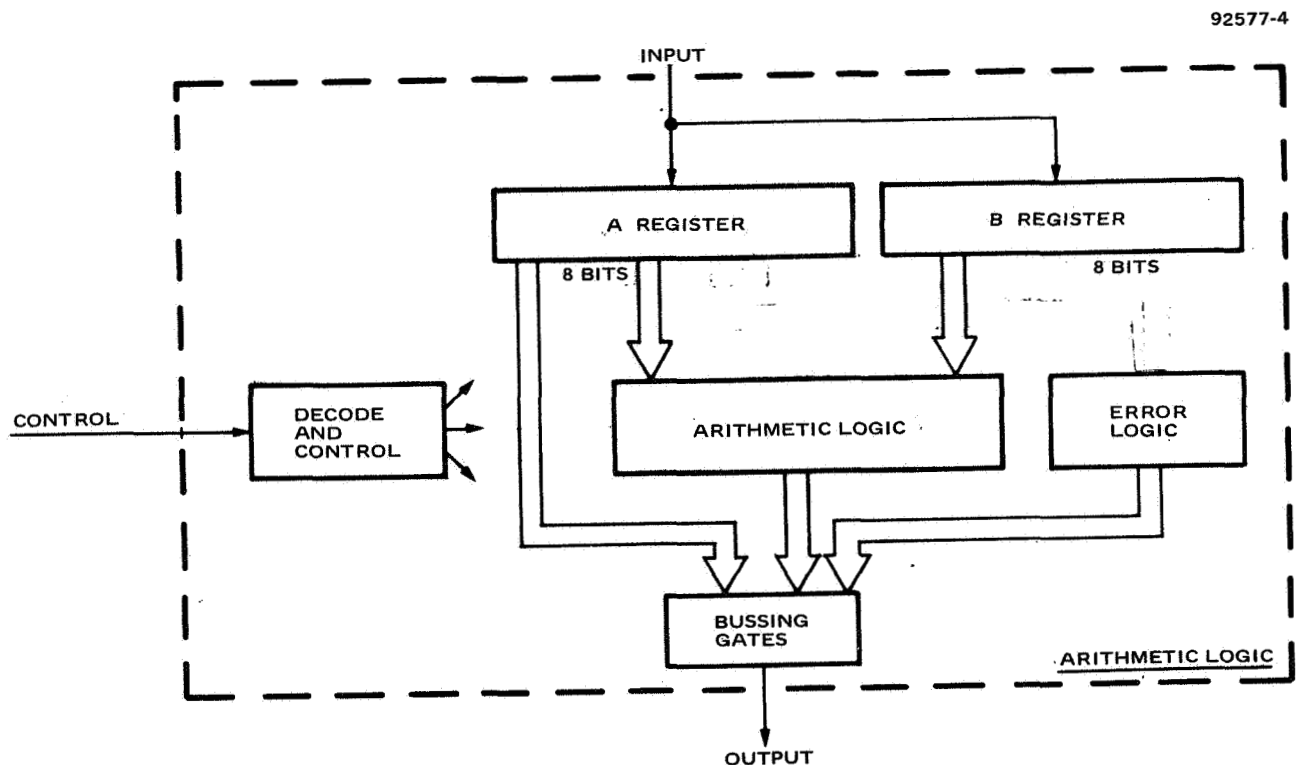


Figure 4. L2 Character Block Diagram

A typical arithmetic operation using the L2 character might proceed as follows: (1) first operands are transferred to A and B registers (2) after appropriate delay results are presented at the output of the L2 character. The error logic provides overflow and carry-out information.

L3 Character

The L3 character provides input/output interface. The L3 character provides input gating for external devices, output gating to a common I/O bus, and an address and a data register. Figure 5 is a block diagram of L3.

92577-5

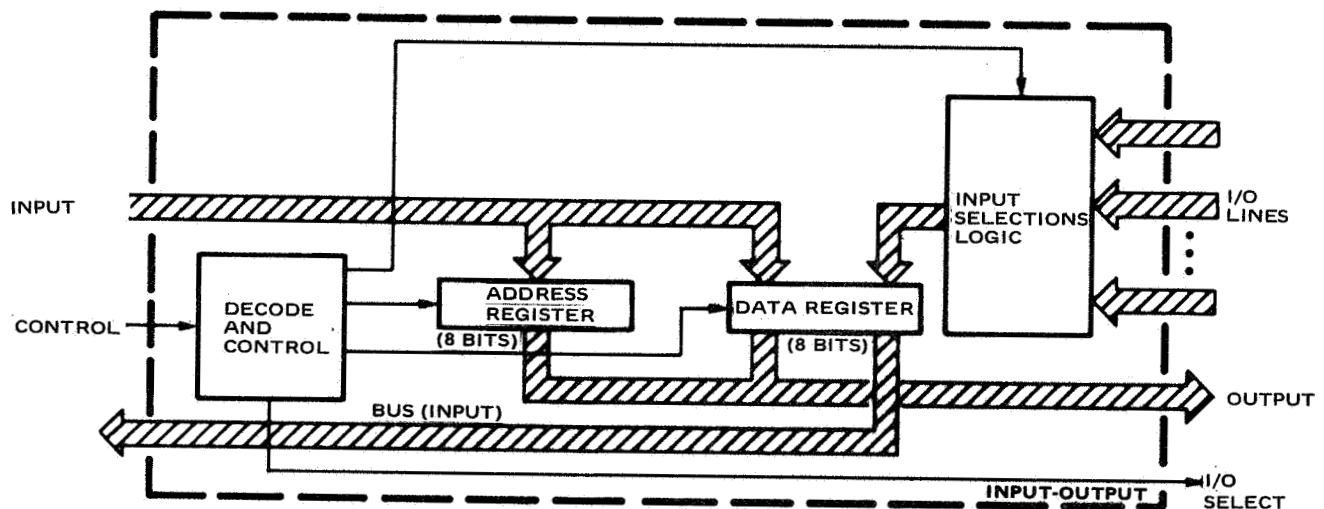


Figure 5. L3 Character — I/O Interface Block Diagram

To input data, an input line is selected under micro-program control resulting in selected data setting the Data Register. From the Data Register, the I/O data may be accessed for micro-program use. To output data, the micro-memory selects the desired I/O device via the I/O select lines. After I/O selection, either (still under micro-memory control) the content of the data register only or the content of the address register followed by the content of the data register after an appropriate delay, appears at the output. I/O device addressing is thus accomplished directly from the micro-memory via the I/O select lines while data in the address register references addresses within the selected device as with main memory.

M1 Character

The M1 character provides the micro-memory address register and related functions. The 10 address bits of M1 allow for addressing up to 1024 micro-memory words. The address is contained in the MMC (Micro Memory Counter) register and serves to address the micro-memory proper. The MMC register contains the address of the next micro-instruction word to be accessed. Associated with the MMC register is an incrementer which automatically steps through micro-program address states. This steps the micro-program sequentially until the program issues a transfer command. This transfer command takes the program out of the present sequence. Figure 6 shows the block diagram for M1.

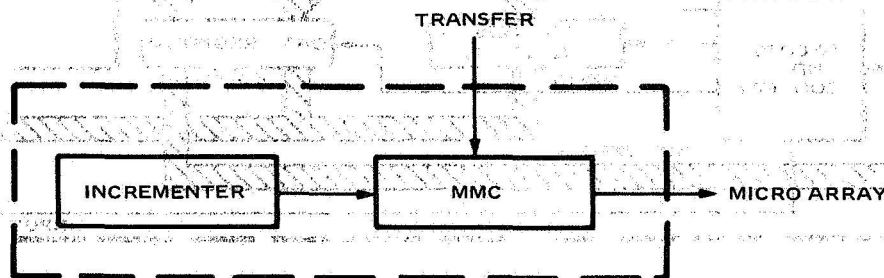


Figure 6. M1 Character Block Diagram

Branching or transferring within the micro-program is provided for by two modes: Unconditional transfer and conditional transfers.

M2 Character

The M2 character contains a micro-memory register. The micro-memory word is divided into three fields. The first and the second fields are instructions and the third is a constant. Timing is derived from the timing base. Figure 7 shows the block diagram of M2.

92577-7

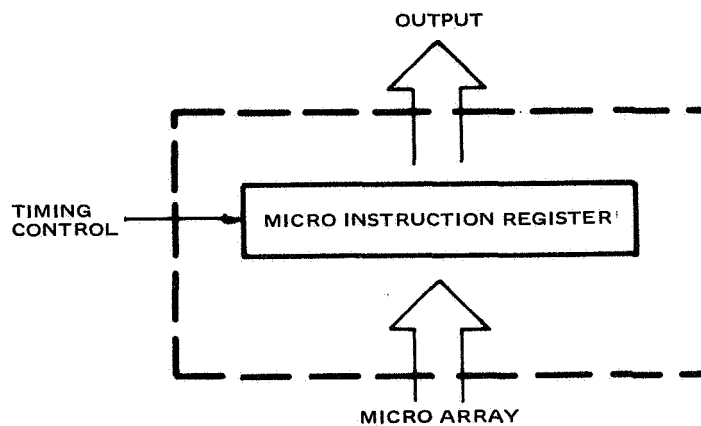


Figure 7. M2 Character Block Diagram

MM Character

The MM character contains the micro-memory array. The address register and word register for the array are located on M1 and M2 respectively. MM is a read-only array. The presence of an address on the input lines causes the contents of referenced location to appear on the output lines after an appropriate delay and to remain there until the input address is removed. The MM character contains 256 word segments. Figure 8 shows the block diagram of MM.

92577-8

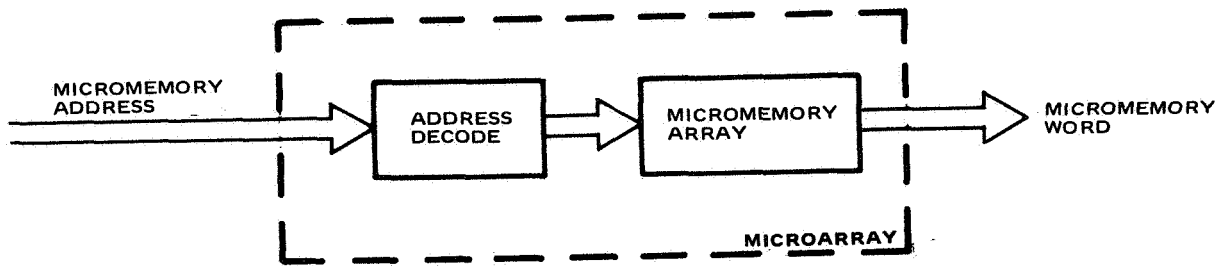


Figure 8. MM Character Block Diagram

P1 Character

The P1 character is a scratch pad memory of 256 bits of storage. The scratch pad is arranged into 16 registers of 16 bits each. Figure 9 is a block diagram of P1.

92577-9

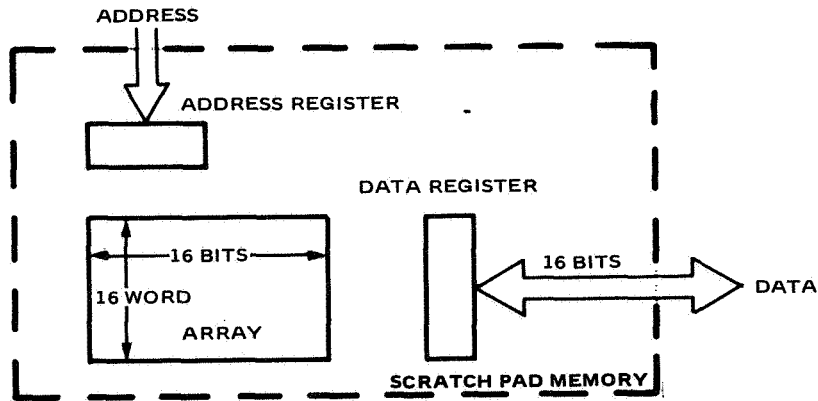


Figure 9. P1 Character Block Diagram

P2 Character

The P2 character is an expandable 8-bit up/down counter with byte look-ahead logic. The introduction of a time signal produces a real-time binary clock. The counter may be read in parallel and is resettable to any desired value. Figure 10 shows the block diagram detail.

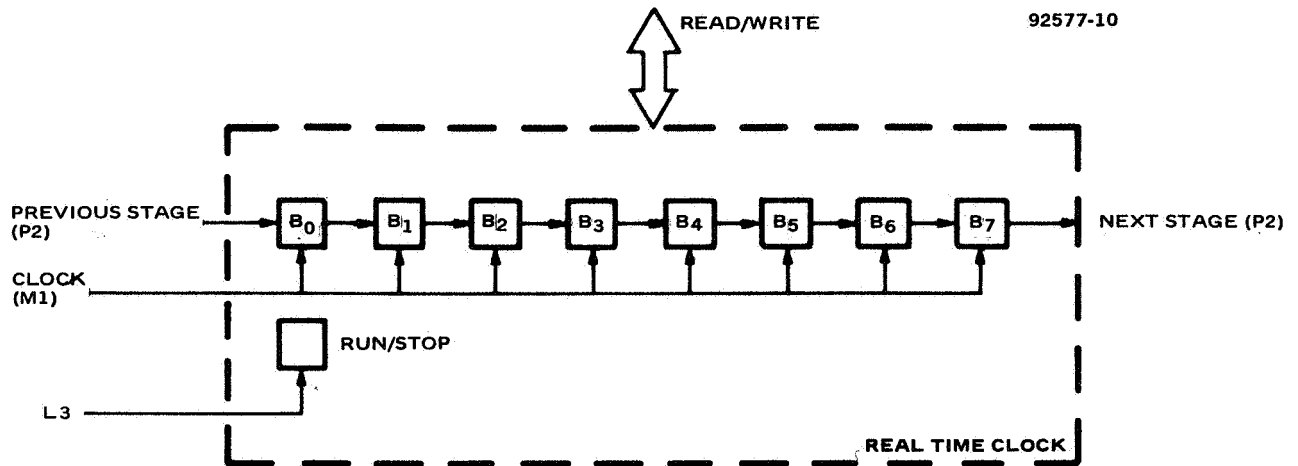


Figure 10. P2 — Character Block Diagram

The P2 character contains control logic allowing the counter to be in a run state or stop state dependent upon micro-program control.

P3 Character

The P3 character provides the capability of switching any three input channels to any three output channels. A 16-bit width is provided. This configuration allows three simplex simultaneous connections. Figure 11 shows the block diagram for the switch.

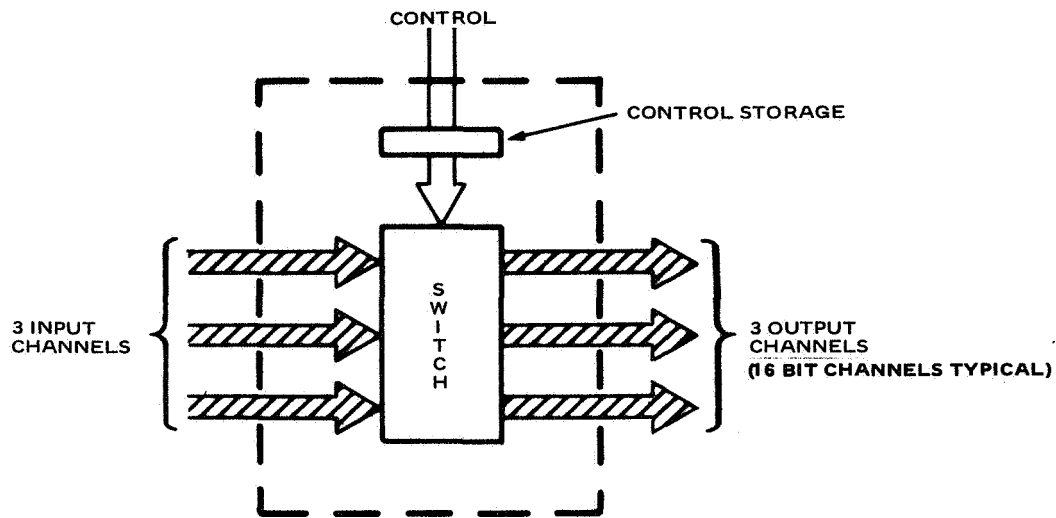


Figure 11. P3 — Character Block Diagram

The input and output channels of P3 may be connected to any electrically compatible interfaces.

DESCRIPTION OF MICROPROGRAM REPERTOIRE

To fully utilize the building blocks of the previous section to implement the design of a digital system, a suitable micro-program repertoire of instructions is required. The repertoire is related to the scope of the logical functions to be performed and the amount of direction that can be conveyed to these logical functions within the contents of one micro-memory word.

Micro-memory Word-Bits

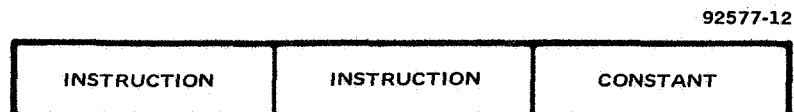


Figure 12. Micromemory Half-Word

A micro-memory word is composed of three fields - - two instruction fields and a constant field. (See Figure 12) The instructions can access the constant field, introducing into the data stream this constant from the micro-memory.

Instruction Fields - The instruction fields are divided into three subfields - source, operator, and destination subfields as shown in Figure 13.

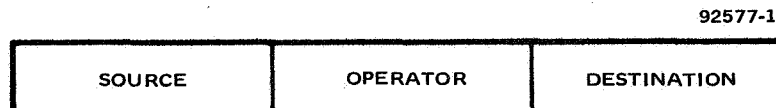


Figure 13. Instruction Field

The source specifies the origin of the data to be operated upon as defined by the operator field. The destination specifies the location where the data result will be stored after the operation is performed.

Source Subfield - The source subfield specifies the source of information for the micro-command. In most cases, the information selected by the source subfield is operated upon as specified by the operator subfield; however, there are exceptions. Particularly, sources not allowed an operator are those dealing with interlogic unit transfer and the incrementer. Sources are divided into two groups--those that are allowed the full range of operators (Group I) and those that are allowed none (Group II). Table I lists the contents of these groups. Data accessed by Group I appears at the input while data accessed by Group II appears at the output of each character.

TABLE 1
SOURCE MNEMONICS

GROUP I		GROUP II	INC
	G1		
	G2		
	.		
	.		
	.		*ECS
	G16		
	ARS		*ARS
	ADS		*ADS
	R10		*R10
	CNT		

Itemized below are the sources and their description.

GROUP I Sources:

G1 - G16 - The general set of registers providing the bulk of the fastest access storage in the functional system. A particular functional system may have 4, 8, 12, or 16 registers in a standard register unit. Register lengths may vary subject to the following constraints:

1. Because registers are built in subgroups of 4, all 4 registers within one subgroup must be of equal length.
2. Register lengths vary by an integral number of bytes.

ARS - Controls the accesses to the content of the A register in character L2.

ADS - Controls the output of the arithmetic unit. Usually this output is the full sum of the contents of the A and B registers, however, the output may be modified as described earlier. The arithmetic unit requires no initiation but continually computes the result from the present contents of the A and B registers so that a result will be valid after an appropriate delay from the time of entry of last operand into register A or B.

R10 - Controls the accessing of data from the I/O channel specified in the constant field and also causes this data to be set into the D₀ register of the L3 character.

CNT - Addresses the Constant field of the micro-memory word. The contents of this field enter the data stream right justified.

GROUP II Sources:

INC - Addresses the content of the logic register of the L1 character. The incrementer logic is unclocked so that after an appropriate delay the increment output is valid without further signals and remains valid and available for access until the L1 register content is changed.

ARS - Addresses the A register of an alternate logic unit and causes the transfer of this data to the ordering unit. No operator is allowed.

ADS - Addresses the output of the arithmetic unit of an alternate logic unit. All conditions applying to adder access under ADS apply here. The sum is introduced into the output data of the logic unit associated with this instruction. No operator is allowed.

R10 - Addresses the D₀ register of the alternate logic unit and introduces it onto the output data of the logic unit associated with this instruction. No operator is allowed.

ECS - This source code transfers the error code of the alternate logic unit to the output data of the logic unit associated with this instruction. No operator is allowed.

Operator Subfield - The operator subfield specifies the type of operation the micro-instruction invokes. These operators are listed below and operate upon the data from the input bus and present the result at the output bus:

RSO - 31

LSO - 31

COM

AND

Each operator is itemized below along with a description of its function.

RSO - 31 - This operator provides for a Right Shift of the operand from 0 to 31 positions. A shift of zero is equivalent to a straight transfer.

LSO - 31 - This operator provides for a Left Shift of the operand from 0 to 31 positions. A shift of zero is equivalent to a transfer.

COM - The Complement Operator produces the ones complement of the operand.

AND - The "AND" operator generates the bit by bit logical AND product of the operand and the contents of the logic register of the L1 character. The contents of the logic register must have been set in a previous cycle time. If the logic register is set in the same cycle time the results are undefined.

Destination Subfield - The Destination Subfield specifies directly the register to receive the instruction result. These register designations are listed below:

<u>Present Logic Unit</u>	<u>Alternate Logic Unit</u>
G1 - G16	
L ₁ RD	
A ₁ RD	*A ₁ RD
B ₁ RD	
MMC	
P ₁ IO	*P ₁ IO
A ₁ IO	*A ₁ IO
S ₁ IO	*S ₁ IO
W ₁ IO	*W ₁ IO
L ₁ IO	*L ₁ IO
AIC	

G1 - G16 - Addresses the general set of registers providing the bulk of fast access storage for the Functional System. These are the same registers as described under the Source Subfield heading.

L₁RD - Address results to be set into the logic register of the L₁ character. The content of the logic register is not predictable if the operator was "AND" or the source was "INC".

A₁RD - Addresses the A register of the L₂ character.

B₁RD - Addresses the B register of the L₂ character.

MMC - The Micro-memory Counter destination provides for a transfer within micro-memory dependent upon the input data.

P₁IO - The Prepare I/O addresses the A₀ register of the L₃ character. No external I/O action is initiated.

A₁IO - The Access I/O addresses the A₀ register of the L₃ character. In addition the I/O peripheral specified by bits in the constant field is signaled to initiate an access cycle using as an address the content of A.

S₁IO - The Store I/O provides for storage of the D₀ register at the I/O addressed in the constant field. Data on the output bus enters A which specifies the address within the referenced I/O.

W₁IO - The Write I/O destination provides for storage of information at the I/O specified in the constant field. Also the data on the output bus enters the D₀ register. The content of the A₀ register specifies the address within the referenced I/O.

LIO - The Load I/O destination provides for storage of information into the D₀ register of the L3 card. No I/O action is initiated.

AIC - Alter Instruction Code allows the micro-instruction exactly one word time away to be altered by the present micro-instruction. The lower order bits of the output data bus are transferred to the micro-instruction register according to mask bits in the constant field. The first bit of the group of three controls alteration of the source subfield, the second bit controls the operator subfield, and the third controls the destination subfield. The mask bits when set allow alteration of their respective subfields.

*ARD, *PIO, *AIO, *SIO, *WIO, and *LIO - Each of these destinations is equivalent to its corresponding destination without the asterisk except that the destination is in the alternate logic unit while the source of information and place of operation is within the logic unit associated with the specifying instruction.

Transfer Field - The transfer field allows for microprogram specification of both conditional and unconditional transfers within the microprogram. At all times when a transfer is not effected (either conditional or unconditional) the micro-memory counter is incremented by one.

There are basically three testable functions. They are: (1) least significant bit - - true; (2) most significant bit - - true, and (3) all bits - false (true = 1, false = 0).

There exist twelve conditional transfer test combinations and one unconditional transfer. The mnemonics used are defined as L = least significant data bit true; M = most significant data bit true; Z = all data bits false; I = data tested at input to logic unit; O = data tested at output of logic unit; and TRA = unconditional transfer.

The following defines each transfer in detail:

LI - Tests the least significant bit for the "one" state on the input to the logic unit.

LO - Tests the least significant bit for the "one" state on the output from the logic unit.

LIO - Tests the least significant bit for the "one" state both on the input and output to/from the Logic Unit.

MI - Tests the most significant bit for the "one" state on the input to the Logic Unit.

MO - Tests the most significant bit for the "one" state on the output to the Logic Unit.

MIO - Tests the most significant bit for the "one" state both on the input and output to/from the Logic Unit.

LIMI - Tests the least significant bit and the most significant bit for the "one" state on the input to the Logic Unit.

LIMO - Tests the least significant bit for the "one" state on the input to the Logic Unit and the most significant bit on the output from the Logic Unit.

MILO - Tests the most significant bit on the input to the Logic Unit and the least significant bit on the output from the Logic Unit.

ZO - Tests all bits on the output for zero from the Logic Unit.

ZOLI - Tests all bits on the output from the Logic Unit for zero and the least significant bit on the input for the "one" state.

ZOMI - Tests all bits on the output from the Logic Unit for zero and the most significant bit on the input bus for the "one" state.

TRA - Ten bits of the micro-memory word are transferred to MMC causing a microprogram jump.

PRELIMINARY IMPLEMENTATION OF THE MCB COMPUTER

The purpose of this section is to emulate the existing design of the referenced NASA MCB Computer as a test of the generality of the functional character set. This presentation is the result of a preliminary analysis of the MCB (Ref. 1 & 2) and a "first pass" at a design using functional characters. (See Figure 14)

General Design Considerations

All aspects of the computer have not been considered. Design remains to be done on the parity generating and checking circuits, input-output and internal discretes, internal error detection and corresponding communication to the Configuration Assignment Unit (CAU). Work on the data block transfers remains incomplete although the loading of the Local Data Memory is designed. It is estimated that the incompleeted portion represents less than 10% of the total computer.

Following is Hughes interpretation of the MCB, representing the major portion of the computer. A preliminary design of the following has been completed. Any known omissions or deviations from the original MCB design have been noted.

Control Unit CU - - The CU controls the operation of the MCB sub-computer and the accessing of main-program instructions. Each instruction is decoded in the CU and all effective address calculations and accessing of data are performed. If the instruction is one that is to be performed in the Arithmetic Unit (AU), the CU transfers the operation field of the instruction and the operands to the AU. If the instruction is one involving main memory loads or stores, program branches, or input-output functions, it is performed internally to the CU. The CU stores the answer (if the instruction is one that produces an answer) in the location specified by the instruction.

The CU is provided with a local data memory (IDM) capable of holding 64 words of data and the contents of the three index registers. This memory is loaded in blocks from the MU and data is transferred back to main memory in blocks, both by means of a special program instruction. Access to this unit by the CU is considerably faster than to main memory. Use of the IDM makes it possible to avoid accessing main memory each time an instruction is executed.

The CU also has a local program memory (LPM) similar in construction to the IDM but containing instructions rather than data. In normal operation, the CU tracks its progress by storing in fast-access registers the address of the present instruction in both the LPM and the main memory. The instruction itself is accessed from the LPM. The LPM acts as a buffer between main program memory and the CU and must be loaded in a fashion that will increase the execution rates by negating the main program memory access time. Efficient loading of the LPM has not been considered in this design. On the flow chart, the LPM is loaded by a block transfer similar to the IDM.

22

(Preliminary study indicates that the micro-programmed functional-logic approach may eliminate the requirement for a local program memory (LPM) without sacrificing the benefits.) Any program branch instruction initiates or changes the sequence of the LPM loading.

The control unit maintains contact with each of the other four units of the sub-computer as follows. (See Table II for sub-routines.)

(a) Arithmetic Unit AU

The CU sends some of the operation codes and associated operands to the AU. It responds to AU interrupts and accepts the resultants from the AU.

(b) Input-Output Unit I/O

The CU sends the I/O units commands and timing signals enabling the I/O unit to establish a data path between the MU and any input or output device. Data flow is in blocks; no data passes through the CU. During the initial design pass of the MCB it was desired to keep microprograms as simple as possible. This led to an input-output block transfer scheme in which the CU tracks the main memory address and word count, and initializes both the memory unit and the I/O unit for each word of the block. It is felt that all of the above are more properly functions of the I/O unit. One of the first modifications of the initial design will be to alter the I/O block transfer so that the CU need only initialize the I/O unit with the device number, starting address, and word count. The I/O will be programmed to access the MU and perform the entire block transfer independently of the CU. A proper memory access priority will be established.

(c) Memory Unit MU

The CU can read or write into main memory one word at a time or in blocks. The CU initializes the MU with a command and address, and after an appropriate delay, accepts or sends the data to the MU. The CU also commands the MU to send or receive data from the I/O unit.

(d) Configuration Assignment Unit CAU

The communication between the CU and the CAU differs somewhat from the original(MCB) design. The CU is capable of writing into or reading those registers and counters assigned to it but physically contained in the CAU (mask registers, diagnostic clocks, cross communication registers, etc.) without disturbing the flow of operations of the CAU. The CU remains attentive to CAU interrupts and responds to commands from the CAU to initiate diagnostics. The CU may transmit a new configuration to the CAU by means of a CAU interrupt.

TABLE II
LIST OF CONTROL UNIT SUBROUTINES

Mnemonic	No.	
EA3	1	Calculates effective address (appropriate index register contents plus contents of const. field of instruction = address) when 3 are needed.
EA1	2	Calculates effective address when only one is needed (access a portion of 1).
RAC2	3	Given an address at random (except one in main memory) it accesses the appropriate storage device and reads in the data. Access 2 address.
RAC1	4	Similar to 3 except only the data from 1 address is called in.
RAW	5	Writes data into any address desired.
AU NPT	6)	
INT AU	7)	used for communication with the AU.
IO NPT	8)	
INT IO	9)	used for communication with the I/O.
CAU NPT	10)	
INT CAU	11)	used for communication with the CAU.
INT MUA	12)	
INT MU TO MU	13)	
INT MU TO MU	14)	used for communication with the MU.

Arithmetic Unit AU - - The arithmetic unit receives the current instruction operation field and the actual operands from the control unit. It decodes the operator field, performs the operation accordingly, and transfers the resultant back to the control unit. The AU has the responsibility to perform all floating point operations, shifts, multiplies, and divides. It has the capability to perform fixed-point add and subtract and all logical operations. The AU also has the capability to detect overflow from shifts and arithmetic operations.

Memory Unit MU - - Each memory unit contains one main memory module capable of storing 4,096 words. The address and data registers associated with the memory module are planned to be read-write registers addressable by the MU microprogram similar to the G registers. The data to and from these registers follows the usual data paths of the characters.

The memory unit interfaces with the control unit and input/output unit. It is capable of transmitting data to and from either unit, acting on commands from the CU. The address must be provided by the CU or I/O. For input-output transfers, data flows directly between the MU and I/O units.

Input-Output Unit I/O - - The input-output unit is currently planned with 4 input and 2 output channels with expansion possible. It provides a data path between any of its external devices and the memory unit (for data flow in either direction) upon receipt of the proper commands from the control unit. The initial design has the I/O-MU transfer as one word per CU interrupt, with word count and main memory address tracking the responsibility of the CU.

The I/O unit currently interfaces with the external devices through an 8-bit data interface. This can easily be expanded to the full 32-bits if necessary.

Configuration Assignment Unit CAU - - The CAU continually monitors the idle-time counter and the diagnostic-time counter. If either is permitted by the CU to go to zero, the CAU immediately takes over the reconfiguration duties. After configuration, the CAU notifies the appropriate CU by means of an interrupt that it is to perform a self-check diagnostic. If this fails, the CAU tries another configuration. (The initial design of the CAU does not go into great detail with respect to diagnostic initiation.) The CAU also has the duty of comparing mask and status registers. If a corresponding "one" is detected, it notifies the appropriate CU to start a diagnostic which may eventually arrive at a new configuration. The CAU accepts this new configuration from the CU and implements it. Then it initiates the self-check in the new subcomputer. If this test fails, the CAU takes over the complete reconfiguration responsibilities.

The part of the CU in the process is as follows. Upon receipt of the command to initiate a diagnostic, it stores the address of the present instruction, branches to the main memory location of the first statement of the diagnostic, and resets the idle-time counter and diagnostic-time counter. If the diagnostic is successful it stops the diagnostic-time counter. If any error

occurs it drops into an idle mode immediately, letting the diagnostic counter go to zero which allows the CAU to take over. At the end of a successful diagnostic the CU either notifies the CAU or resumes operation if the diagnostic was self-initiated rather than CAU-initiated.

The CAU contains 16 registers. Various other units may read or write directly into these registers. (See Figure 24) The versatility of the G1 character makes it possible for any unit to write or read its assigned registers without calling attention of the CAU. This is possible because the CU-CAU interface is not restricted to 8 bits. The CAU also contains 4 counters which are similar to the registers in their accessing and address characteristics. The CAU has physical control over the byte interface switches. These switches are designed to be addressable and to accept coded state information and act accordingly.

The CAU is capable of setting the status register according to information received from discretes from any unit or by CU interrupt.

Control Unit Design

This topic contains the control unit block diagram, microprogram flow chart

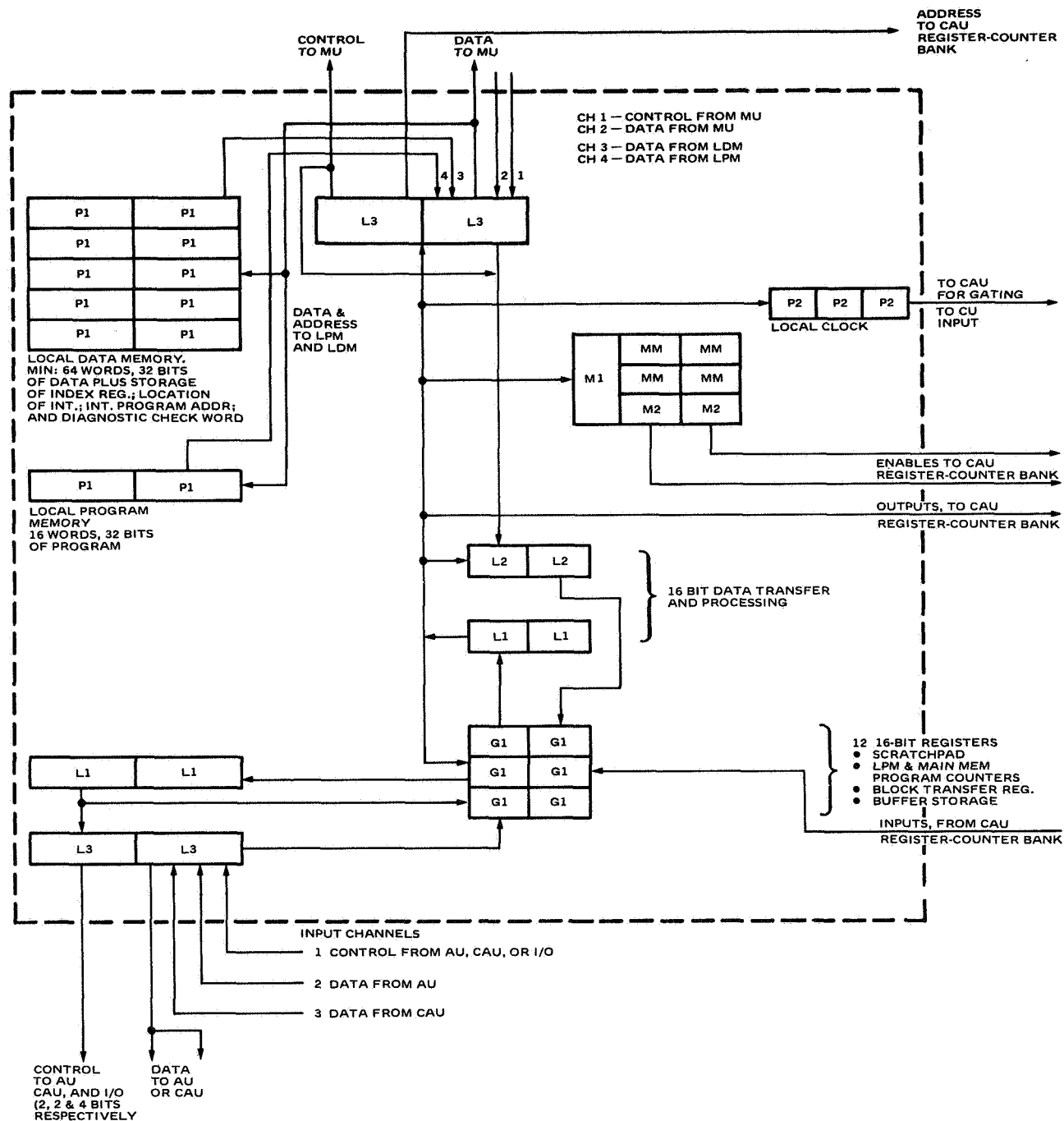
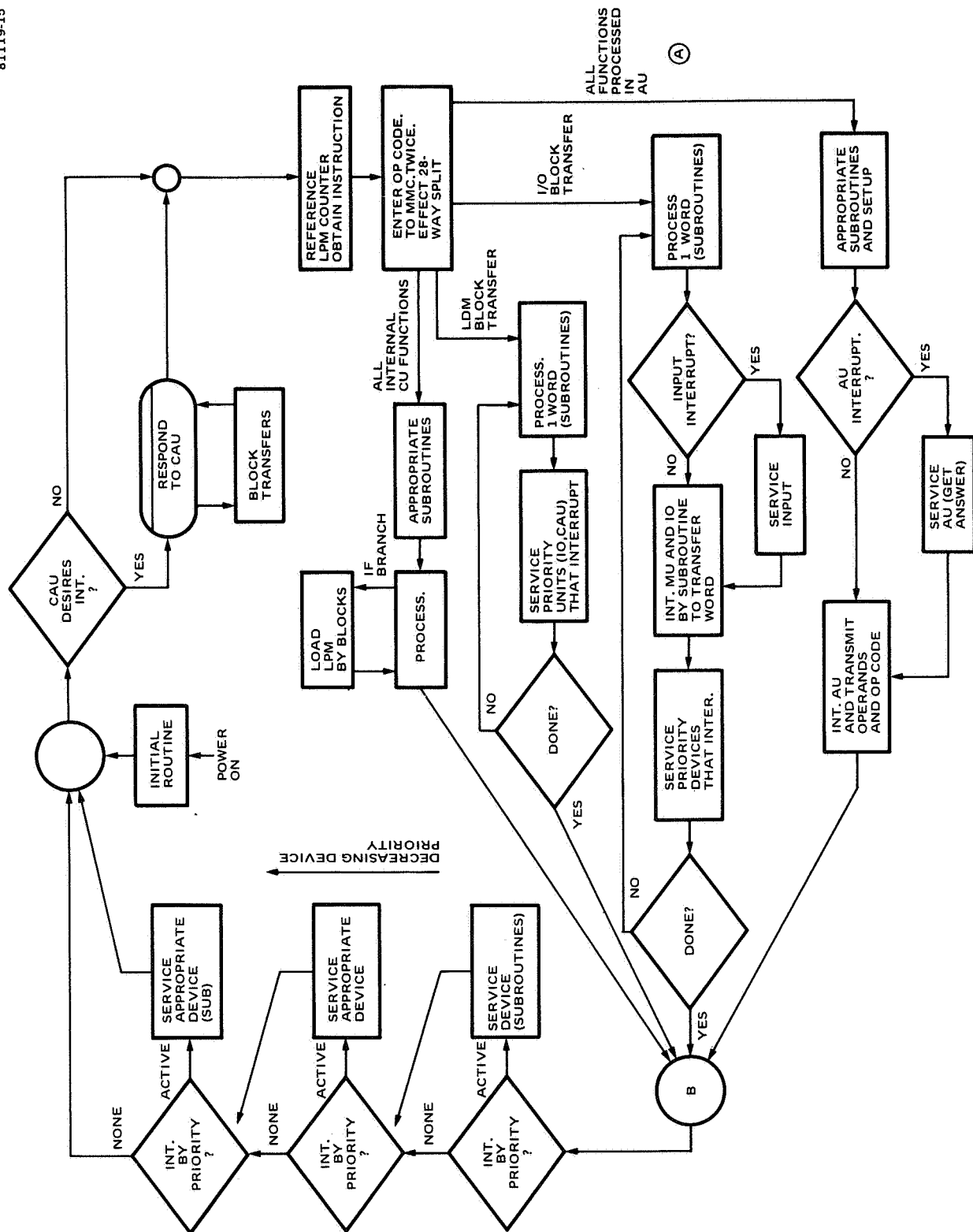


Figure 15. MCB Control Unit — CU — Block Diagram



NOTE: ("DIRECT ADD", BETWEEN POINTS (A) AND (B) ON THE CU MAIN FLOW CHART)

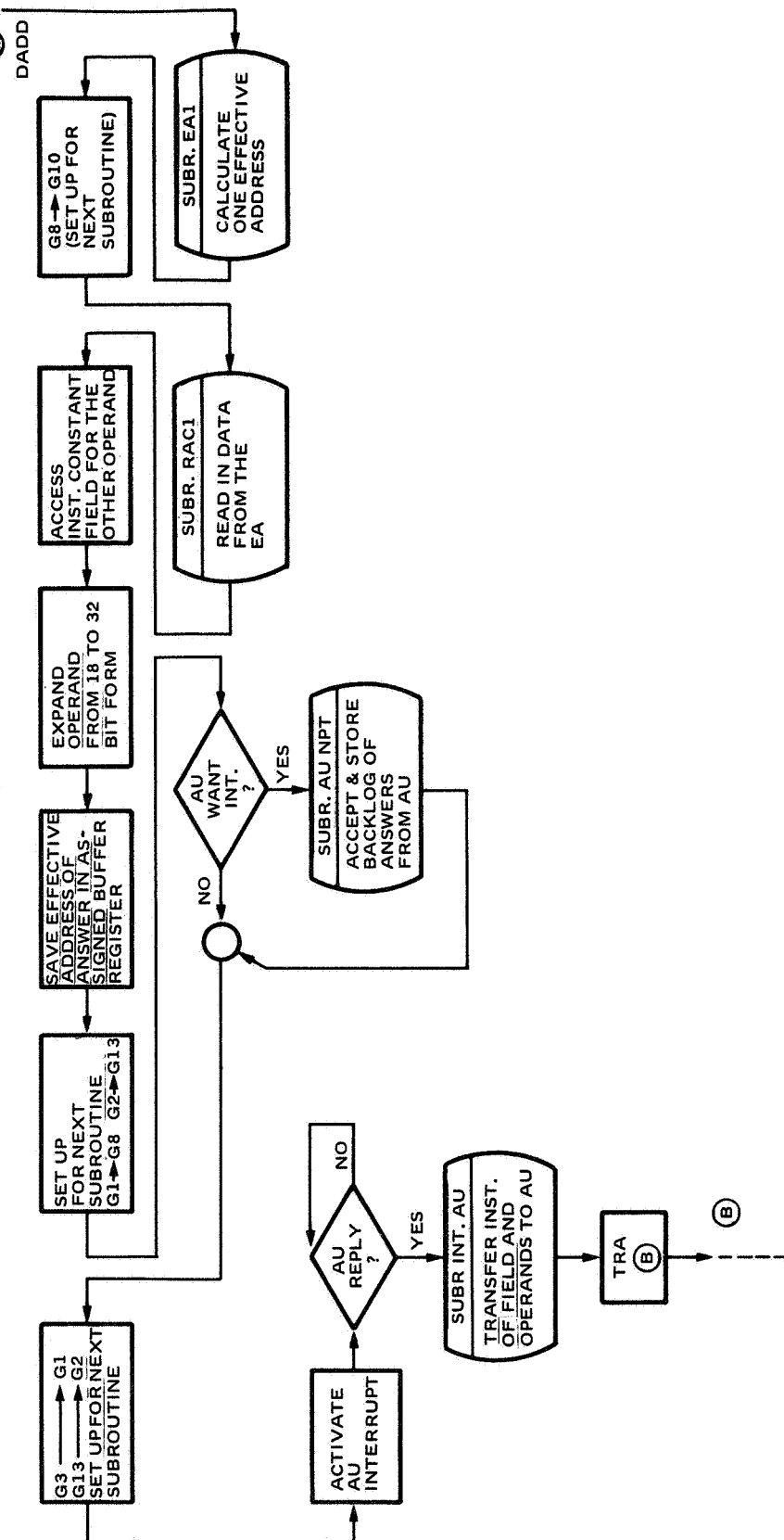


Figure 17. Control Unit Microprogram Flow Chart — Detailed Example

Arithmetic Unit Design

This topic contains the arithmetic unit block diagram, and microprogram flow chart. (See Figures 18 and 19)

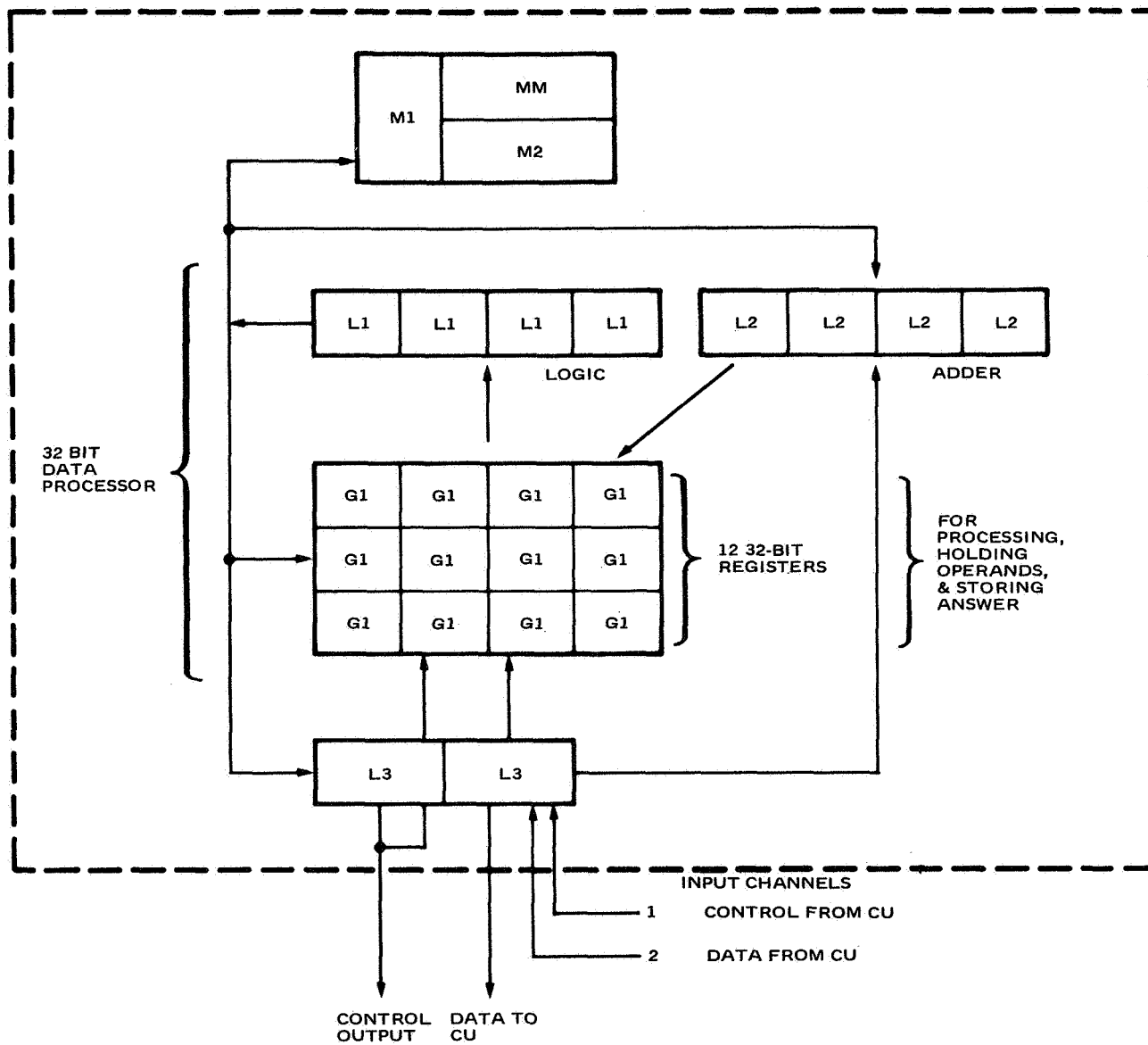


Figure 18. MCB Arithmetic Unit — AU Block Diagram

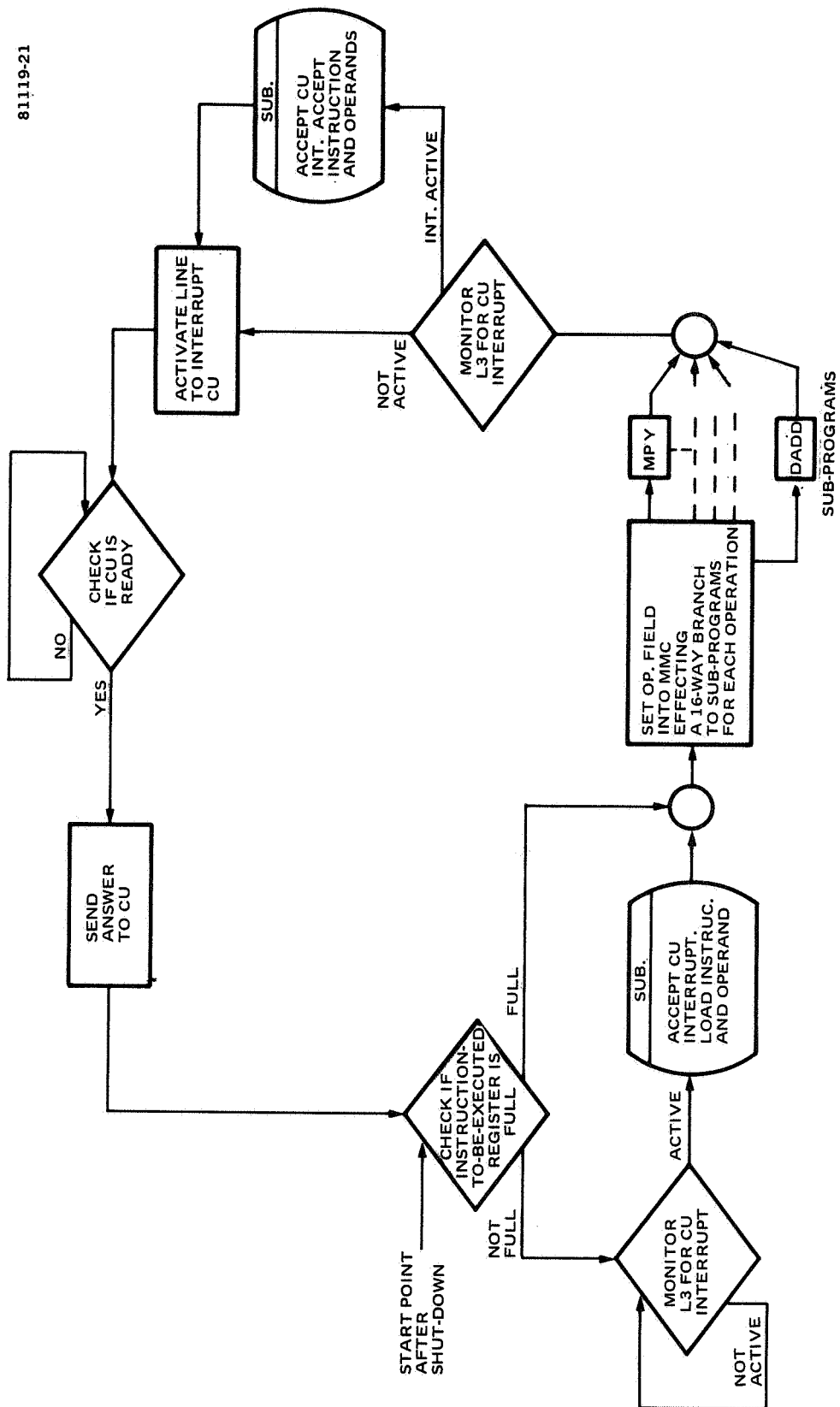


Figure 19. Arithmetic Unit Microprogram Flow Chart

Memory Unit Design

This topic contains the memory unit block diagram, and microprogram flow chart. The MU has the smallest microprogram which is relatively trivial. The examples show the two subroutines which transmit data to the I/O or CU. This also illustrates a special application of the L3 character in which no device number needs to be given. (See Figures 20 and 21)

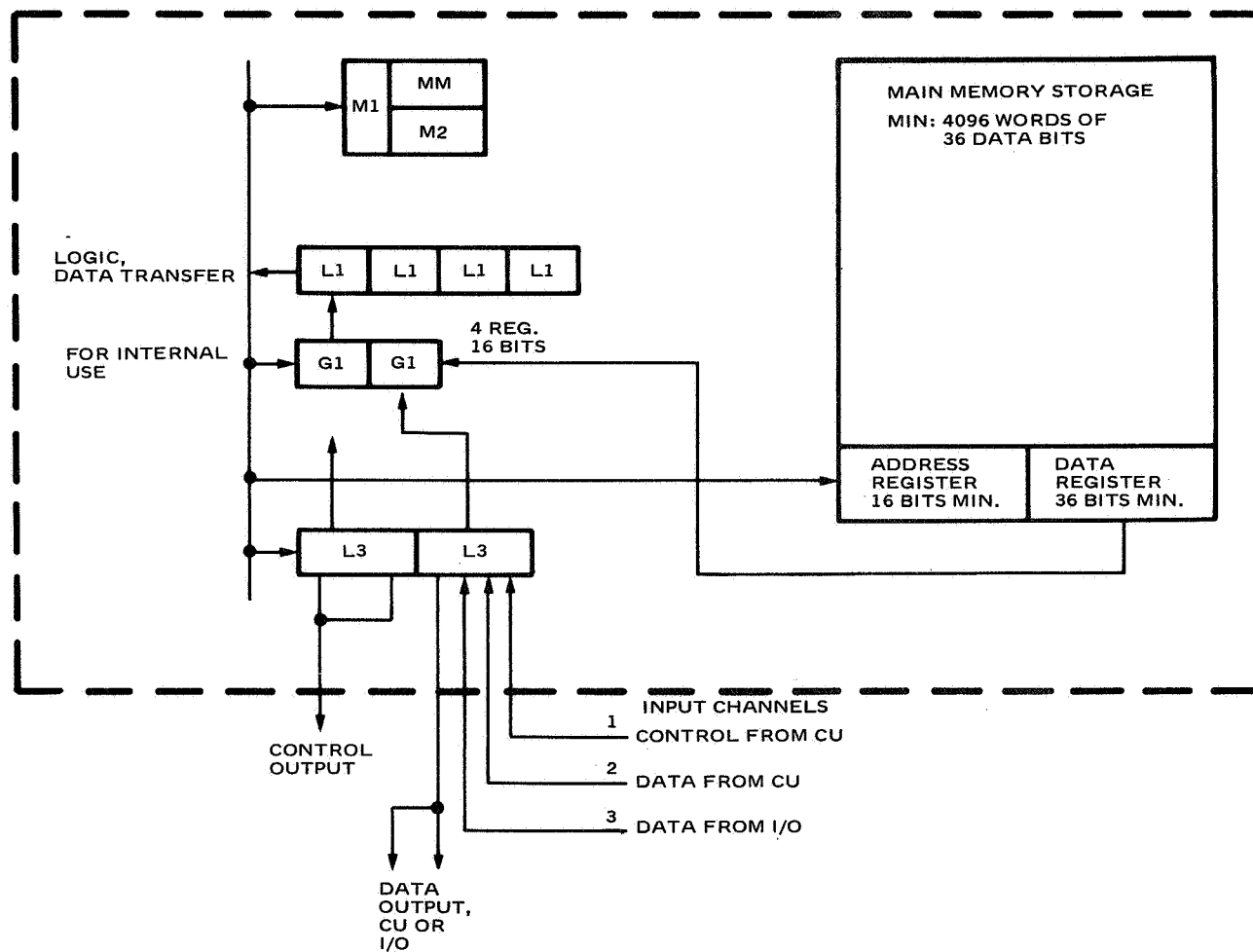


Figure 20. MCB Memory Unit — MU — Block Diagram

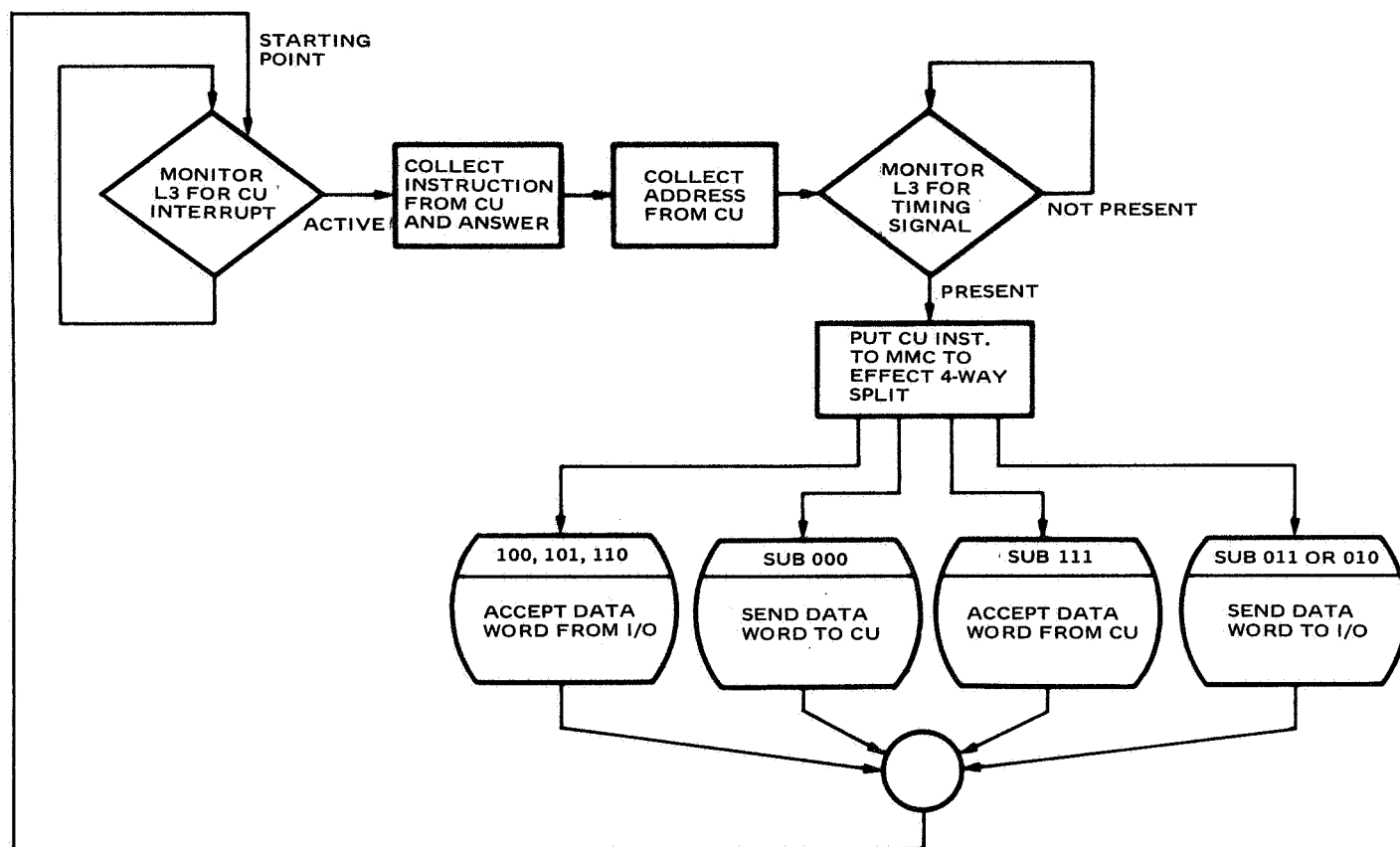


Figure 21. Memory Unit Microprogram Flow Chart

Input-Output Unit Design

This topic contains the input-output unit block diagram, and microprogram flow chart (See Figures 22 and 23)

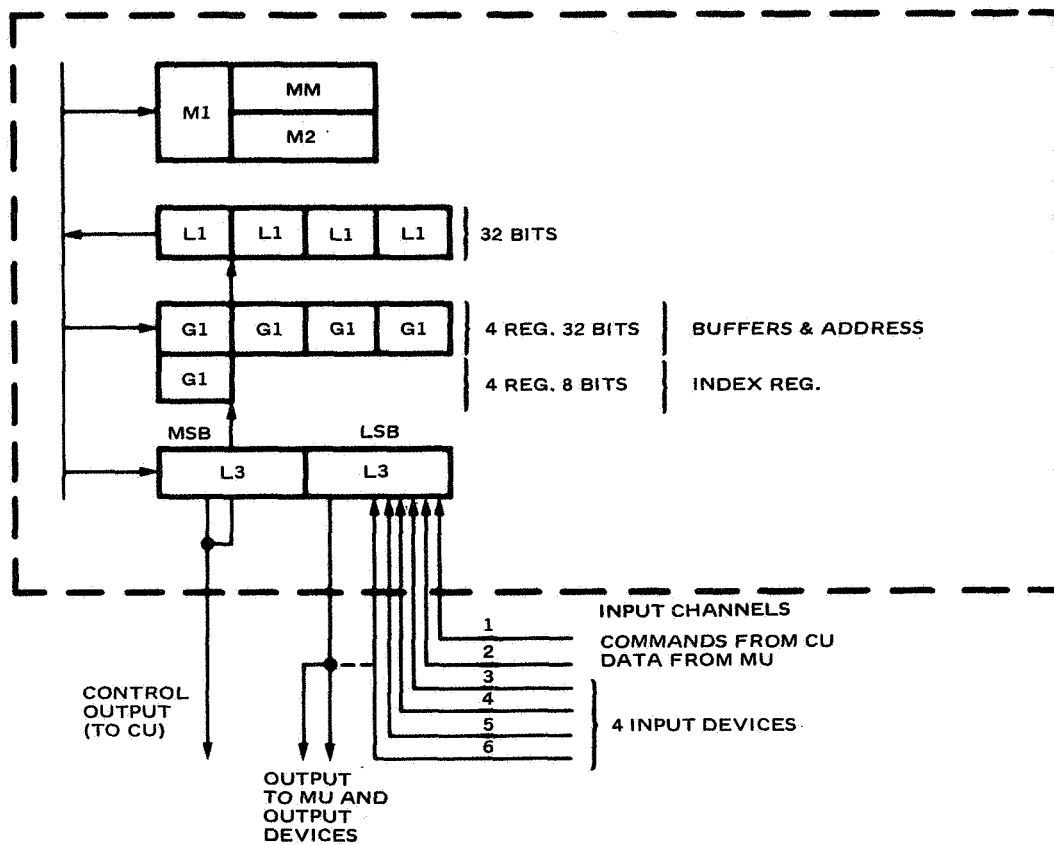


Figure 22. MCB Input-Output Unit — I/O Block Diagram

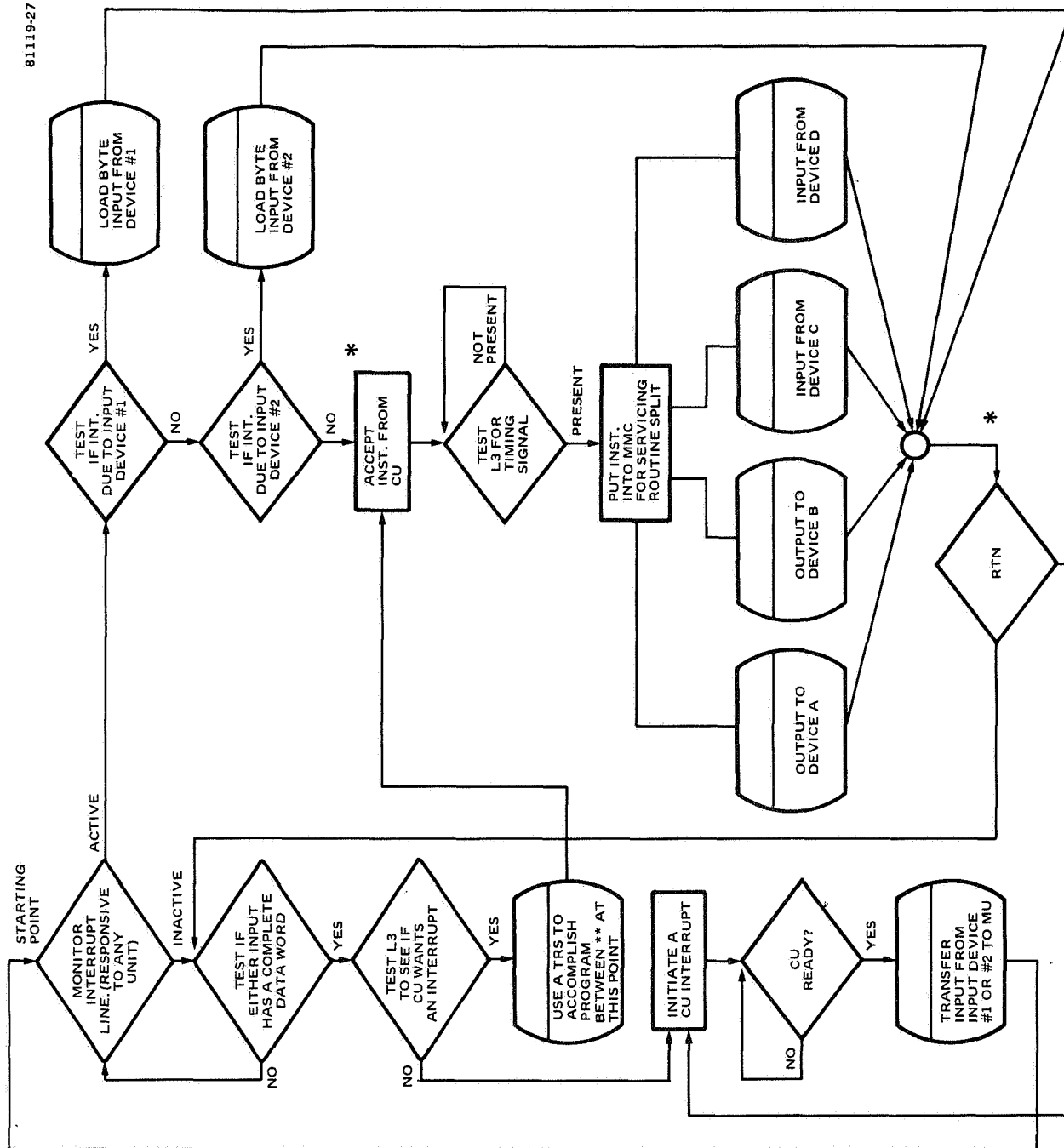


Figure 23. Input-Output Unit Microprogram Flow Chart

Configuration Assignment Unit Design

This topic contains the configuration assignment unit block diagram, and microprogram flow chart. The example is of the statements making up the central flow of the CAU. The CAU has been initially designed to operate with a relatively simple microprogram. (See Figures 24 and 25).

81119-29

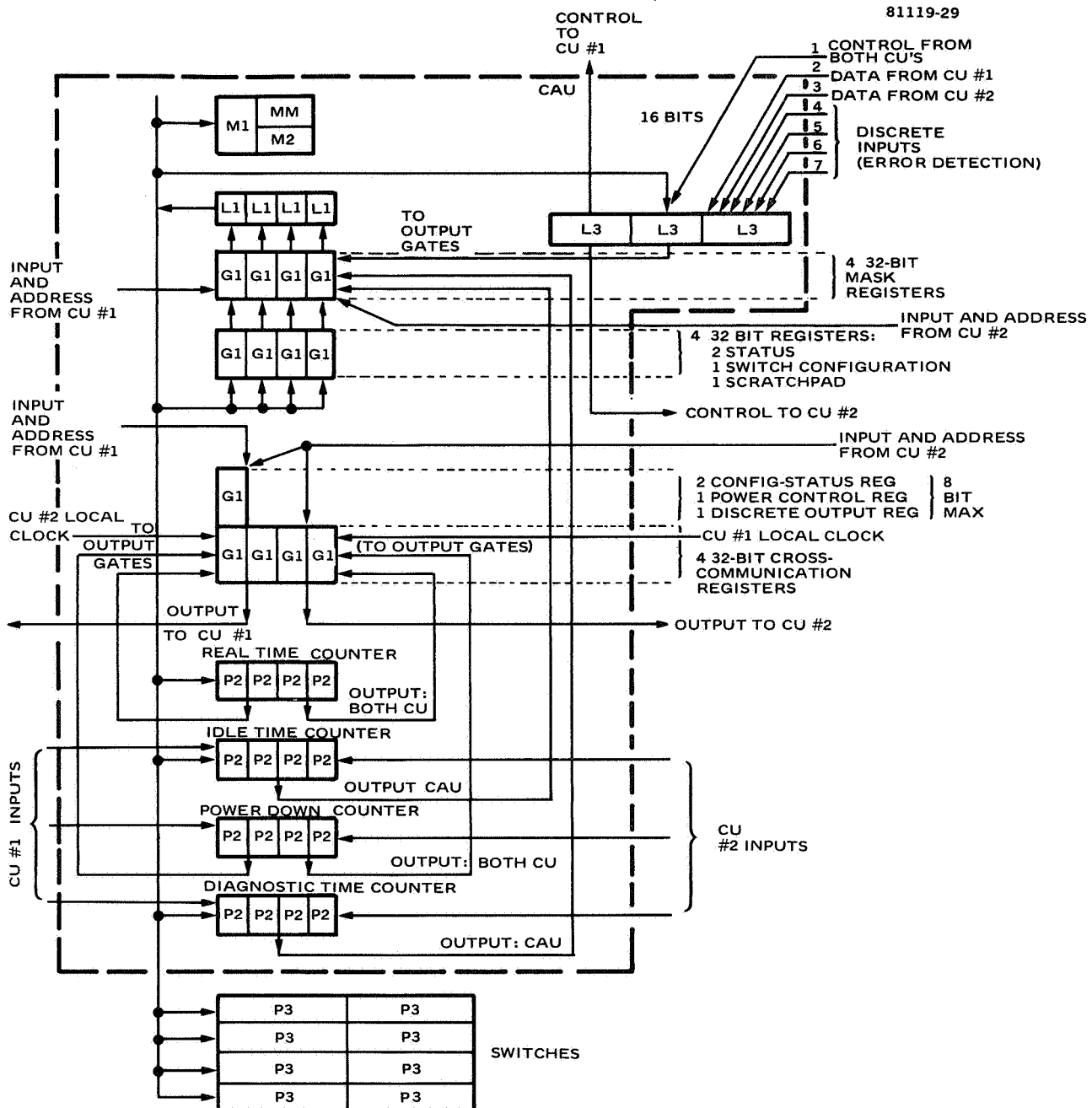


Figure 24. MCB Configuration Assignment Unit — CAU — Block Diagram

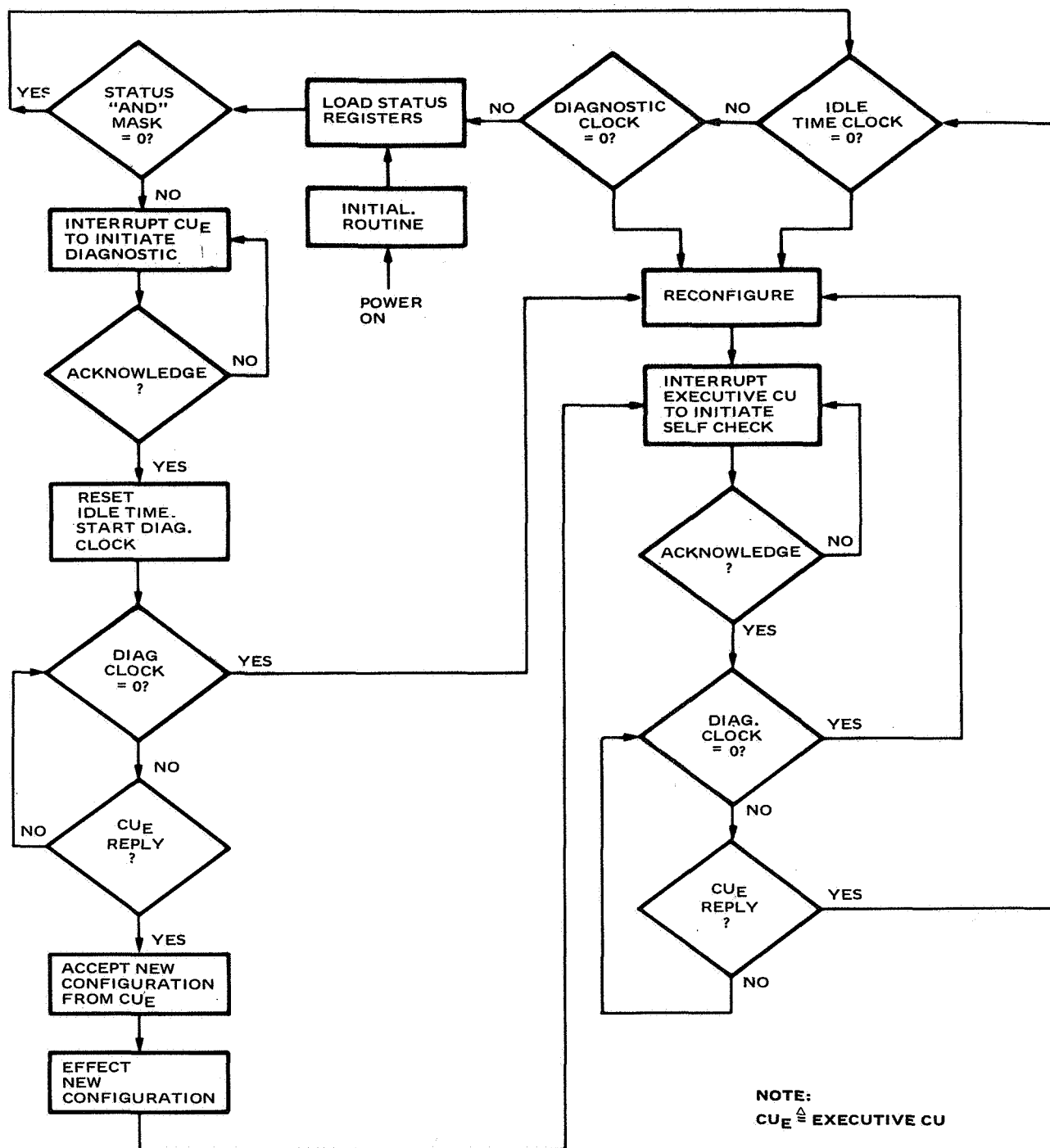


Figure 25. Configuration Assignment Unit Microprogram Flow Chart

MCB EXECUTION RATES

This section contains a list of those operations for which execution times have been determined (Table III). Floating point equalization and normalization times are included. It is stressed that these are times resulting from the first-pass design in which no strenuous effort was taken to minimize them.

Because of the generality of the Hughes character set, the control unit has the capability to perform certain of the more simple arithmetic operations. Permitting the CU to perform them (rather than the AU as in the existing MCB design) relieves the burden on both units. The CU in the existing design spends more time communicating with the AU than in doing those operations itself. To implement this modification means simply to alter the CU micro-program; no change in the hardware is required. If this modification were implemented, the pertinent revised execution times would be as shown in Table

TABLE III
MCB EXECUTION TIMES

	<u>Hughes Prelim.</u>	<u>MCB Existing</u>
ALL SHIFTS	12.2us	14.5us
ADD DIRECT (FIXED POINT)	9.9	11.0
ADD "	11.6	11.0
SUBTRACT "	11.6	11.0
MULTIPLY, MOST SIG. HALF (FIXED POINT)	20.1	29.0
MULTIPLY, LEAST SIG. HALF (FIXED POINT)	20.3	29.0
INCLUSIVE OR	11.5	9.5
EXCLUSIVE OR	11.5	10.0
LOGICAL AND	11.5	9.5
FLOATING POINT ADD	15.1	10.5 + Δ
FLOATING POINT SUBTRACT	15.1	10.5 + Δ
FLOATING POINT MULTIPLY	23.6	26.5 + Δ
STORE DIRECT	6.2 + W	5.5 + W
LOAD DIRECT	6.6 + W	5.5 + W
BLOCK TRANSFER SETUP	1.9	1.5
PER WORD	4.4 + W	5.0 + W
INPUT OUTPUT (AS ORIGINALLY DONE) SETUP	1.9	4.5
PER WORD	4.1 + W	7.0 + W
BRANCH DIRECT	6.6 + W	2.0 + W
CONDITIONAL BRANCH	7.2 + W	4.0 + W
DECREMENT AND BRANCH	7.6 + W	3.0 + W
BRANCH AND LINK	9.1 + W	5.0 + W

Δ - equalization + normalization time

W - memory wait time

TABLE IV
MCB EXECUTION TIMES AFTER MODIFICATION

	<u>Modified*</u> <u>Hughes</u> <u>Prelim.</u>	<u>MCB</u> <u>Existing</u>
FIXED POINT DIRECT ADD	6.0 μ s	11.0 μ s
FIXED POINT ADD	7.6	11.0
FIXED POINT SUBTRACT	7.7	11.0
INCLUSIVE OR	6.8	9.5
EXCLUSIVE OR	7.3	10.0
LOGICAL AND	6.9	9.5

* Modification allows CU to perform selected operations instead of AU.

EVALUATION OF PRELIMINARY IMPLEMENTATION OF MCB

In evaluating the preliminary implementation of the modular computer breadboard (MCB) using functional characters, the following factors must be considered:

- Feasibility of logic design technique
 - Adequacy of functional characters
 - Adequacy of microprogram instruction repertoire
 - Ease and practicability of microprogramming a complete digital system
- Number of Functional characters and total quantity
- Speed of operation
- Power, size and weight
- Reliability
- Cost
- Other (See Section on "Weighted Design Considerations".)

At this point in the study, no meaningful conclusions can be reached on some of these factors. They are in fact, heavily dependent on the particular hardware selected for the implementation of the computer. Clearly, implementation of the MCB computer with the functional characters will give significant advantages in most areas when LSI techniques are utilized. The minimization techniques of the discrete circuit era are not applicable to LSI circuits. During the transition period from discretes to LSI (at this time) the LSI values and tastes have not yet been developed and therefore confusion exists in the technical community regarding these values. Also at present the potential benefits of LSI cost and reliability improvements are not being realized as fast as was expected. Care and patience must therefore be the watchword when evaluating LSI circuits vs discrete implementations.

Considerations

Feasibility of Logic-Design Technique - - The adequacy of the functional characters and microprogram instructions quickly became evident when implementing the MCB. Having defined the units of the MCB based on available information, the logic designer knowledgeable in the available functional characters and micro-instruction repertoire, readily coded the necessary microprograms for all units. As the design proceeded, it became readily apparent that all system functions of the MCB units could be completely defined using the functional

characters and proper microprograms. No apparent modification of the defined microprogram repertoire seems required for efficient design and only minor changes to one functional character is planned at this time.

A significant development during this study was the discovery that a reduction in the type of units (AU, MU, CU, CAU, and I/O - see preceding section) of the MCB becomes practical using the functional character - microprogramming technique. Due to the flexibility of the technique, many of the functions reserved for the AU in the MCB design are readily handled by the CU resulting in significant hardware reduction as well as eliminating some inter-unit communication problems. This development should be more fully explored, but clearly is beyond the scope of this effort.

Types of Functional Characters and Total Quantity - - Ten basic types of functional characters are utilized in implementing the MCB computer. This is in contrast with the types of cards in the present MCB implementation. (See Section on "Description of Building Blocks"). The MM character actually has eight variations resulting in a total of 17 configurations in an implemented MCB. However potential exists for the fabrication of one MM character and program encode the control information by electronic means. As described in the previous paragraphs, no serious modifications appear warranted and, in fact, the versatility of the characters can result in a reduction of computer units. It appears that given the system requirements of the MCB computer, one could arrive at an original implementation for the computer with significantly less total hardware count. The preliminary functional character implementation of the MCB resulted in using 223 characters vs. 554 cards in the present MCB.

Speed of Operation - - The speed of instruction execution of the MCB computer, as implemented with functional characters, compares favorably overall with the original specifications. (See Section on MCB Execution Rates). As explained hereinbefore, the execution times can be improved with a system redesign to more efficiently utilize the character implementation. During the second half of the study some time will be devoted to refining the microprograms. A reduction in number of microprogram steps will decrease computer instruction operating times.

The preliminary instruction operating times were determined assuming a 20 MHz clock rate for logic and a 10 MHz rate for micromemories. This clock rate is feasible under worst case analysis.

Power, Size and Weight - - These parameters of the character-implemented MCB computer cannot be determined until a selection of circuit type has been made. The section of this report on "Weighted Design Considerations" gives emphasis to factors affecting the selection of circuits.

Reliability - - The reader is referred to the Section on "Weighted Design Considerations" for a general discussion on reliability for aerospace computers. No attempt to determine reliability of the character-implemented MCB can be made until a circuit-type is selected.

Conclusions

The defined functional character set and microprogram instruction repertoire are adequate to perform the logic design of the MCB computer. A reasonable number of characters are required to implement the MCB computer. The MCB operation compares favorably with a version implemented with "conventional" logic. The versatility of the functional characters suggests more efficient computer organization of the same system design can be achieved and therefore can result in a faster machine with less hardware. Further, analysis and modification of microprograms can produce reduction in microprograms steps and contribute to improved operating speeds.

SPECIAL-PURPOSE COMPUTERS

Special purpose computers treated in this section are DDA's and serial-to-parallel converters.

Digital Differential Analyzers (DDA)

The solution to DDA logic partitioning is a straightforward extension of the techniques and equipment of the general-purpose computer partitioning. The best method depends heavily on the extent to which these functions occur in the overall system. As this is not yet specified, several possible solutions are discussed.

1. If DDA's are uncommon and the frequency of inputs to them low enough, the modular guidance and control (G/C) computer may be fast enough to justify incorporation of DDA functions into its regular program. To this end, two extra inputs to the I/O unit of the MCB have been provided and the I/O unit itself designed to collect these slow inputs at whatever rate is desired. More inputs are available to the I/O unit if necessary. However, priority interrupts to the G/C computer may cause loss of data if DDA functions are ignored for too long a time.
2. It is possible to implement a small, special-purpose computer to perform DDA functions. The general-purpose computer characters MM, M1, M2, I1, I2, I3, and G1 are used and no new characters are introduced. The justification for introducing an entire computer system to perform DDA functions is as follows. If the computer system is truly modular, it becomes possible to build a computer as large or as small as desired. Therefore a system can be built to match the lesser requirements of the DDA units and still be economical with respect to cost, size, weight, and power.

The main consideration is the number of DDA's necessary to justify the use of a micromemory character group. A preliminary calculation indicates that it may be practical to implement a single DDA consisting of approximately 6 integrators (Figure 26) using one micromemory group. The resulting unit would be capable of accepting data from an A→D converter, perform all DDA functions, and input data to the I/O unit of the MCB. A DDA with 32-bit data registers is implemented with 28 characters. (Figure 27) Approximately 32 program statements per integrator are used, leaving 60 statements free for intercommunication. Each integrator is serviced in 2.8 usec, making iterations of the DDA possible at speeds up to 16.8 usec/iteration. (Included is a sample program flow chart for one integrator, Figure 28).

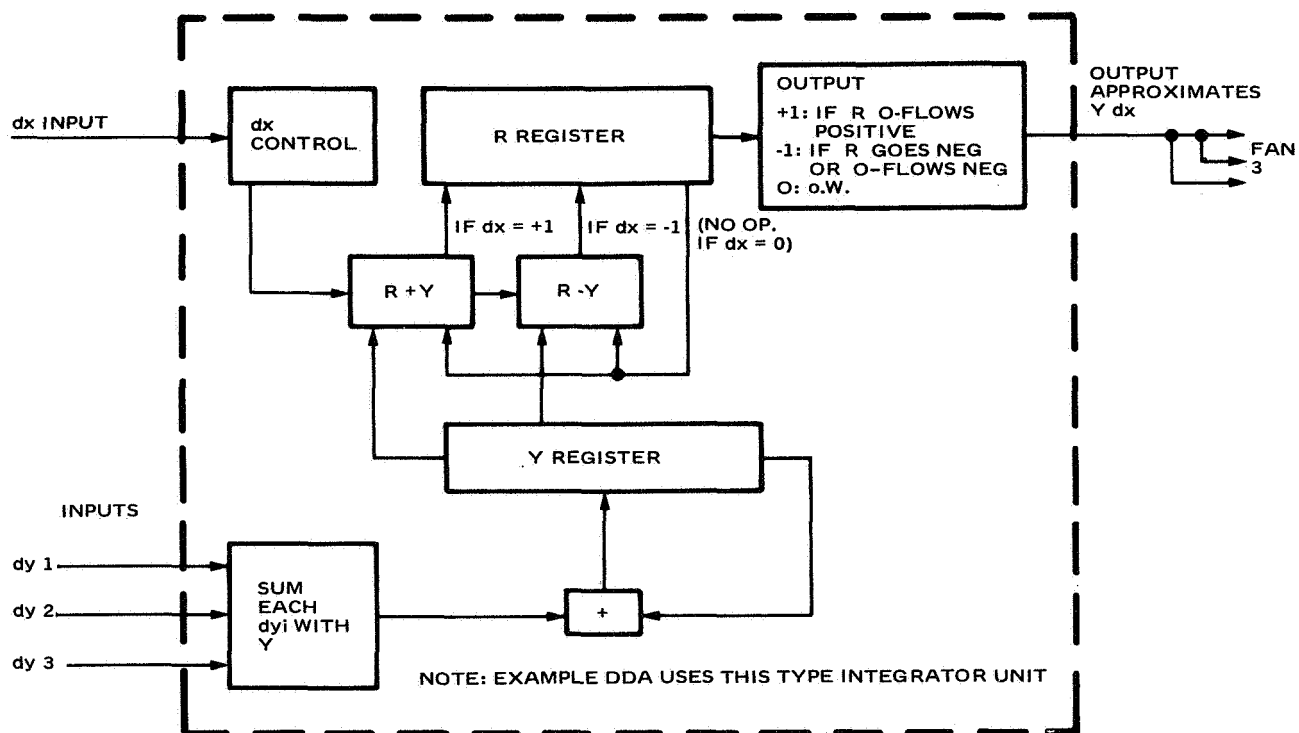


Figure 26. Integrator Unit Block Diagram

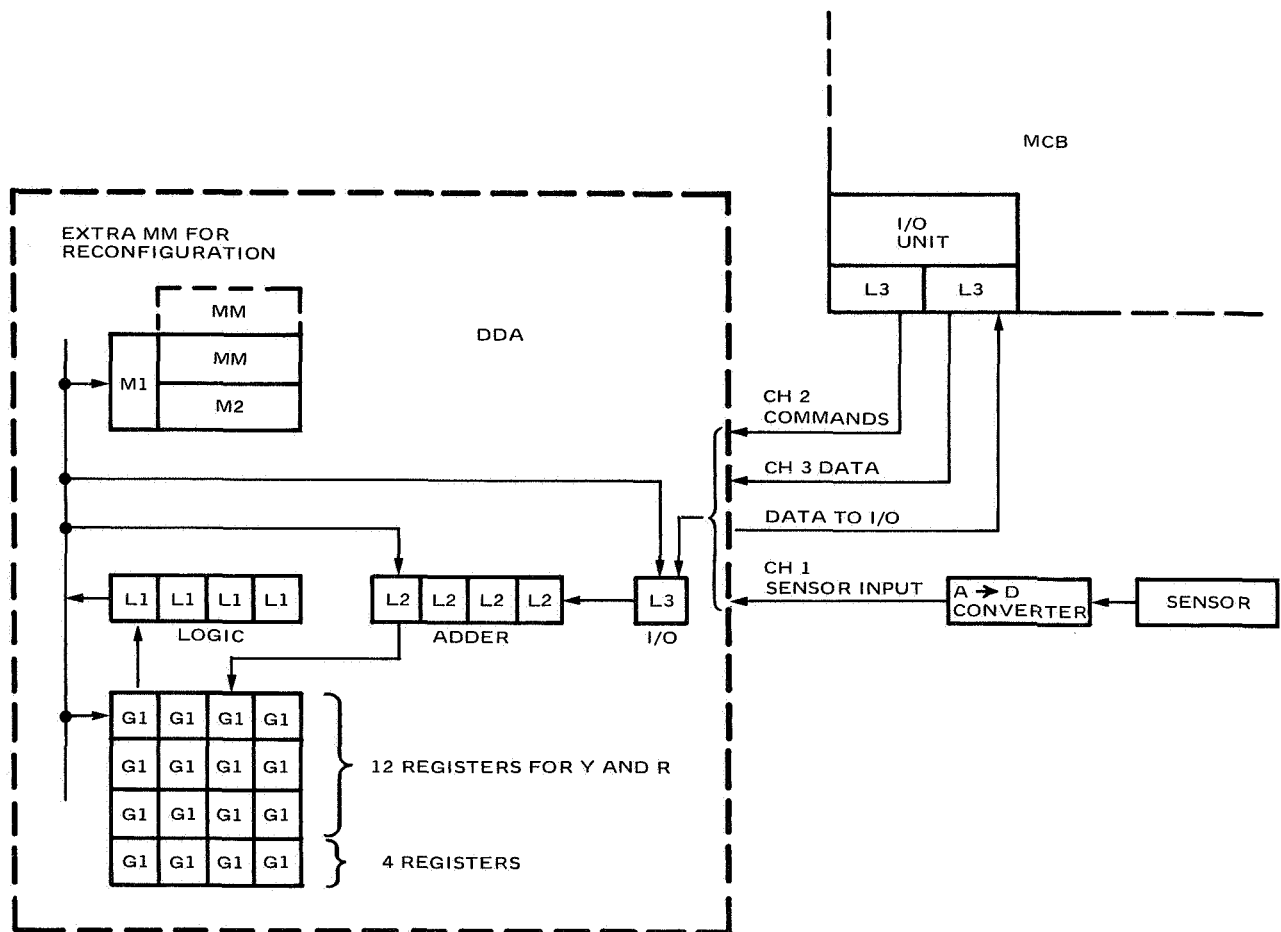


Figure 27. DDA Block Diagram

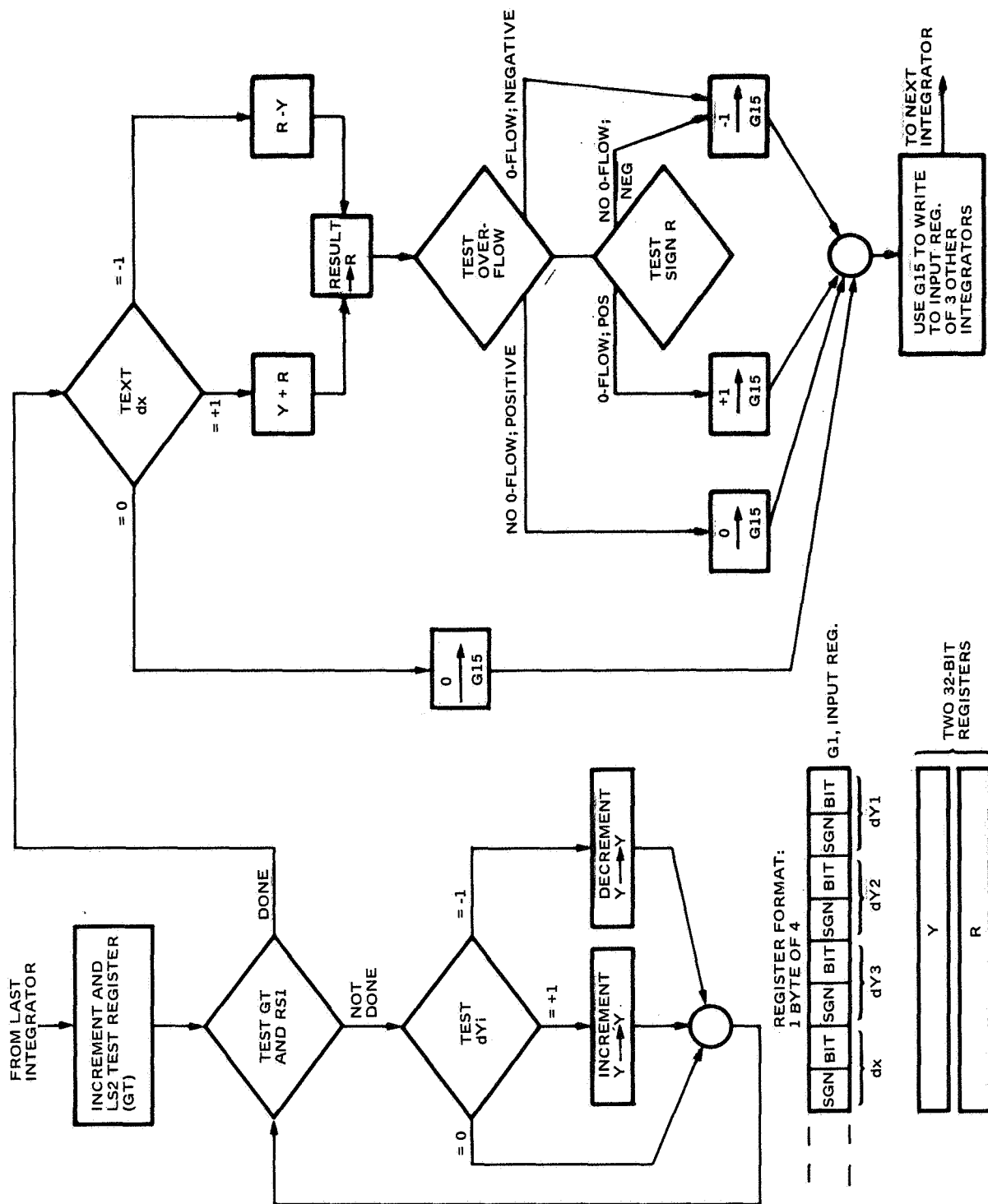


Figure 28. SAMPLE PROGRAM FLOW CHART FOR DDA

Since the micromemory is not alterable, the best solution for reconfiguration of the DDA is to store a different program in micromemory for each configuration, accessible by the main computer through an interrupt to the DDA computer. (An alternate solution is to make the input and output of each integration available for external configuration, but this requires an excessive number of input-output cards.) The main G/C computer also has the requirements of initializing and scaling the DDA's.

3. If it is desired to implement only the integrator units, it is possible to do this using one extra character not yet designed and the general purpose computer characters. None of the memory characters are necessary. However, integrator units of this type are commercially available using less hardware than is possible by adapting the existing characters. This approach must utilize current methods of interconnecting integrators to form a DDA.

Serial-Parallel Conversion

Serial-Parallel and parallel-serial conversion may be treated in the same fashion. Like the DDA, this requires no special equipment or capabilities beyond that available with the general-purpose computer characters.

If the number of conversions is small and performed infrequently, the capability may be programmed in the main computer. If these conversions are common, a special-purpose computer similar to the DDA computer may be constructed to handle them.

Using this method, the bit-rate is somewhat limited by the speed of the micromemory.

SPACEBORNE COMPUTER CHARACTERISTICS

To minimize effort, two previous surveys have been taken as the basis for this compilation of spaceborne computer characteristics. The first is "State of the Art of Aerospace Digital Computers, 1962-1967", by D. O. Baechler (Ref. 3). The second is "Trends in Aerospace Computers" by R. L. Hooper and L. D. Amdahl (Ref. 4). Pre-1965 computers were eliminated from the first compilation. The first 25 entries in Table VI are the remainder. Entries 26 through 35 are computers from the second compilation which were not included in the first. Entries 37 through 39 were suggested by Dr. H. E. Maurer of NASA-ERC. Entry 36 is described in the July 1968 issue of Computer Design. It is intended that this portion of the study shall remain a working report throughout the present study. Therefore, revisions will be made from time to time.

In general, the observations of Baechler about memory capacity, memory cycle time, instruction format, word length, arithmetic type, addressing methods, instruction repertoire and instruction execution times remain true. Since the time of that survey, it has become increasingly clear that in all of these categories the aerospace computer is approaching the ground-based computer in capability. A very significant trend embodied in the IBM 4 Pi family is the use of a subset of instructions from a ground-based computer for the aerospace version. Heretofore, programming has represented a large fraction of the developmental cost of an aerospace computer. The commonality of instruction sets promises to reduce the programming effort. Simulators for program checkout will no longer be required. In addition, a ground-based programmer can easily be converted to an aerospace-computer programmer. Other benefits are the potential use of assemblers and compilers which are designed for the ground-based computer.

A significant trend in aerospace computer characteristics is reliability improvement through the use of hardware. The computers listed in Table VI generally use bipolar integrated circuits. The IBM Saturn IB/V LVDC with a TMR organization is the first of three computers directed primarily toward reliability improvement. A second computer, the JPL-STAR, is a research vehicle which embodies several redundant techniques. The most unusual of these is product encoding of data. Other encoding schemes employed are n-out-of-m for control words and residue encoding of memory addresses. In addition to the encoding redundancy, switching and massive redundancy are used.

The United Aircraft MCB is a third computer using redundancy. It uses switching redundancy and has as a goal the use of a minimum number of types of 100-gate integrated circuit modules. While less ambitious with respect to the application of a number of reliability techniques than the STAR computer, the application of new logic-design techniques to achieve the minimum number of integrated circuit types is quite important to the near-future production systems.

Appendices A and B contain the instruction lists for the IBM 4 Pi-EP and Honeywell ALERT computers. They are considered to be state-of-the-art examples.

TABLE V
SPACEBORNE COMPUTER CHARACTERISTICS

NAME DATE INTRODUCED	DATA FLOW	DATA TYPE	COMPUTING TIME, μ SEC			MEMORY						IN/OUTPUT		PHYSICAL CHARACTERISTICS				COMMENTS	
			NO. OF INSTRUCTIONS			TYPE	WORD SIZE (BITS)	CAPACITY (WORDS)		ACCESS TIME (μ SEC)	CYCLE TIME (μ SEC)	NO. OF CHANNELS	NO. OF INTERRUPTS	TYPE OF HARDWARE	WEIGHT (LBS)	SIZE (CU. FT.)	POWER (WATTS)		MTBF (HRS)
			ADD	MULT	DIV			MIN	MAX										
1. LITTON L-304 EARLY 1965	P	Fx 63	5.6	61		DRO CORE	32	4K	131K		1.8	8	1	TTL IC	34	0.26	100	2300	DATA WORD, 16 BITS; INSTRUCTION WORD, 32 BITS
2. NORTONICS NDC-1051 EARLY 1965	P	Fx 51	8	70	176	DRO CORE	24	2K	8K	2	4	1	4	IC	28	0.5	94	8500	WEIGHT, SIZE, POWER AND MTBF ARE GIVEN FOR 2K MEMORY
3. AUTONETICS D26J LATE 1965	P	Fx 27	8	13	18	DRO CORE	12	4K	8K		4	4	2	DTL IC	20	0.21	62	18,000	16-BIT WORD OPTION IS AVAILABLE
4. HONEYWELL ALERT LATE 1965	P	Fx 89	2	12	30	NDRO 51 AX	24	4K	32K	1	4	3	24	HLTTL IC	73	1.2	150	10,000	AVAILABLE OPTIONS INCLUDE DRO CORE WITH 1 μ SEC CYCLE TIME, AND BUFFERED I/O WITH 7 CHANNELS
5. NORTONICS NDC-1051A EARLY 1966	P	Fx 51	6	26	50	DRO CORE	14	8K	32K		2	2	8	DTL IC	38	0.87	225	3060	WEIGHT, SIZE, POWER AND MTBF ARE GIVEN FOR 16K MEMORY
6. SPERRY MARK XII EARLY 1966	P	Fx 13	18	60		DRO CORE	21	6K				1		IC	64	1.5	250		MEMORY CAN BE PARTIALLY HARDWIRED
7. TRW MARCO 4418 EARLY 1966	P	Fx 27	10	70	73	DRO CORE	18	4K	8K		5	1	0	DTL IC	34	0.4	75	20,000	MEMORY CAN BE PARTIALLY HARDWIRED
8. CDC 5360 MID-1966	P	Fx 41	12	90	90	DRO CORE	24	4K	32K	1.5	6	12	1	IC	28	0.6	95	7545	WEIGHT AND SIZE EXCLUDE POWER SUPPLY
9. COMPUTING DEVICES OF CANADA AN/UYK-501 MID 1966	P	Fx 110	8	110	116	DRO CORE	24	4K	32K	1	2	1	64	IC	82	0.96	240	1180	WEIGHT, SIZE POWER AND MTBF ARE GIVEN FOR 8K MEMORY. HAS ALTERABLE MICROPROGRAM
10. SPERRY MARK XIV MID 1966	P	Fx 13	16	60		DRO CORE	21	6K				1		IC	64	1.5	250		MEMORY CAN BE PARTIALLY HARDWIRED
11. TI 2501 MID 1966	P	Fx 36	4	27	37	DRO CORE	32	4K	16K		2	17	32	IC	65	1.1	350		WEIGHT, SIZE AND POWER ARE GIVEN FOR 4K MEMORY
12. UNIVAC 1830-A MID 1966	P	Fx 72	4	20	34	DRO CORE	30	4K	131K		2	32	1	IC	200	2.85	567	4500	ALSO HAS DRO THIN FILM CONTROL MEMORY AND 64-WORD CORE ROPE BOOTSTRAP MEMORY. SUCCESSOR TO 1830
13. AUTONETICS D26C LATE 1966	P	Fx 87	12	45		DRO CORE	30	8K	32K		6	2	8	IC	47	0.65	175	15,600	HIGH-SPEED MEMORY USED FOR SCRATCH PAD

81119-36

TABLE V (CONTINUED)

8119-37

NAME	DATE INTRODUCED	DATA FLOW	DATA TYPE	COMPUTING TIME, μ SEC				MEMORY				IN/OUTPUT		PHYSICAL CHARACTERISTICS				MTBF (HRS)	COMMENTS
				ADD	MULT	DIV	NO. OF INSTRUCTIONS	TYPE	WORD SIZE (BITS)	CAPACITY (WORDS)	ACCESS TIME (μ SEC)	CYCLE TIME (μ SEC)	NO. OF CHANNELS	NO. OF INTERRUPTS	TYPE OF HARDWARE	WEIGHT (LBS)	SIZE (CU. FT.)	POWER (WATTS)	
14. CDC 5400	LATE 1966	P	Fx	73 3.1 25 275				DRO CORE NDRO THIN FILM	24 58	4K 3K		2.5 2.5	5	16	IC	60	1.1	140	4-WORD (86-BIT) READOUT FROM NDRO MEMORY TO AN INSTRUCTION LOOK-AHEAD MEMORY
15. HONEYWELL SIGN III	LATE 1966	P	Fx	53 4 24 24				DRO CORE	20	2K	0.65	2.0	5	1	DTL IC	28	0.49	92	HAS DOUBLE PRECISION ADD INSTRUCTION
16. HUGHES-HCM-205	LATE 1966	P	Fx	42 4 24 25				DRO CORE	18	2K		2.0	1	1	DTL IC	16	0.2	110	WEIGHT, SIZE AND POWER ARE GIVEN FOR 2K MEMORY
17. IBM 4 PI/CP	LATE 1966	P	Fx	30 6.6 26.1				DRO CORE	32	8K 32K	0.9	2.5	3	5	TTL IC	57	.85	250	HAS 1024-WORD MICROPROGRAMMING MEMORY 70 BITS/WORD, 195 MSEC ACCESS TIME. AVAILABLE AS CP-3 WITH HARDWARE DECODE INSTEAD OF MICROPROGRAM.
18. IBM 4 PI/EP	LATE 1966	P	FL	135 5.8 9.5 18.3				DRO CORE	36	8K 128K	0.9	2.5	3	5	TTL IC	62	0.9	303	3K x 100 BITS MICROPROGRAMMING MEMORY FLOATING POINT OPTION, 32-BIT DATA WORD, 1 PARITY BIT PER 6-BIT BYTE
19. MIT BLOCK II AGC	LATE 1966	P	Fx	34 23.4 46.6 91.9				DRO CORE CORE ROPE	16 16	2K 36K		11.4 11.4	15	10	DCTL IC	58	1.0	100	HAS DOUBLE PRECISION ADD.
20. SPERRY MARK XVI	LATE 1966	P	Fx	14 12 34				DRO CORE	21	8K 16K		6	4		IC	60	1.5	250	MEMORY CAN BE PARTIALLY HARD-WIRED
21. CDC 449	EARLY 1967	P	Fx	36 28 604				DRO THIN FILM NDRO μ BIAX	24 256 3540				1	1	IC	12	.083	4.0	INCLUDES BATTERIES, KEYBOARD AND DIAL READOUT, 12-BIT PARALLEL, 2 BYTE SERIAL
22. IBM 4 PI/TC	EARLY 1967	P	Fx	54 15 51				DRO CORE	8	16K		2.5			TTL IC	27	0.48	75	2 BYTE SERIAL 8-BIT PARALLEL MEMORY 2 BYTES/WORD IN TERMS OF 8-BIT BYTES DATA WORD IS 2 BYTES AND INSTRUCTION WORD IS 1.2 OR 3 BYTES
23. UNIVAC 1818	EARLY 1967	P	Fx	28 4 22 22				DRO CORE CORE ROPE	18 18	1K 4K 8K		2 2	12	10	IC	35	0.7	197	WEIGHT, SIZE AND POWER ARE GIVEN FOR 8K MEMORY
24. LITTON L-3050	DEVELOPMENT	P	Fx	52 3.3 6				DRO CORE	32	4K 131K		1.6	1	64	IC	44	.33	130	WEIGHT, SIZE AND POWER ARE GIVEN FOR 8K MEMORY
25. RCA VIC-36A	DEVELOPMENT	P	FL					DRO CORE DRO CORE	38 38	8K 32K 512	.550 .326	3.0 0.6	4	4	IC	120	2.6	325	SOME DISCRETE COMPONENTS USED, VARIABLE INSTRUCTIONS CONTROL MICROPROGRAM
26. CDC 5500A				70 5. 25.					26	4K 16K		2.5				45	0.7	125	8,000

TABLE V (CONTINUED)

81119-38

NAME DATE INTRODUCED	DATA FLOW	DATA TYPE	COMPUTING TIME, μ SEC			MEMORY					IN/OUTPUT		PHYSICAL CHARACTERISTICS				MTBF (HRS)	COMMENTS		
			NO. OF INSTRUCTIONS	ADD	MULT	DIV	TYPE	WORD SIZE (BITS)	CAPACITY (WORDS)		ACCESS TIME (μ SEC)	CYCLE TIME (μ SEC)	NO. OF CHANNELS	NO. OF INTERRUPTS	TYPE OF HARDWARE	WEIGHT (LBS.)			SIZE (CU. FT.)	POWER (WATTS)
									MIN	MAX										
27. GE A224			68	6.	17.			24	4K	262K		3.0				21	0.44	108	10,000	
28. GE A605			142	8.	34.			36	*4K			3.0				36	0.78	150		
29. GP GPK-20			32	20.	100.			20	2K	8K		4.0								
30. GP L-90-3			32	3.3	41.1			28	8K	16K		2.4				35		100	2,000	
31. LITTON LC-728			61	7.2	31.6			38(?)	0.4K	32K		2.2				32.9	0.55	200	4,250	JOVIAL COMPILER
32. RAYTHEON R-11			70	1.9	5.6			24	8K	131K		0.95				100	1.9	500	3,500	JOVIAL COMPILER
33. RAYTHEON R-25			40	6.2	20			24	4K	65K		3.0				38	0.75	390		
34. TELEDINE AAF55			29	12.	40-60			20	1K	16K		5.0				33.5	0.52	250	2,500	
35. UNIVAC 1830A			62	2.	18.			30	32K	131K		2.0				190	2.7	567	4,500	FORTRAN COMPILER
36. RAYTHEON MID 1968			63	2.6	11.4	11.4		23+ PARITY	8K (?)											PROVIDES FOR DUAL MEMORIES
37. JPL STAR DEVELOPMENT			6					32+	4K 16K	48K	(R/W) (ROM)									FAULT DETECTION, RECOVERY AND REPLACEMENT CARRIED OUT BY HARDWARE
38. TELEDYNE CENTAUR	P	FX	27	6.	22.5	40.5	DRO CORE	24	8K 8K	16K	(R/W) (ROM)	3.0	1	5		30	0.5	110		
39. UNITED AIRCRAFT MCB DEVELOPMENT			28	11.	58. ^D			32 + 4 PARITY	4K	64K	1.0									

^a 24 BITS
^b 28 BITS
^c TWO MULTIPLICATIONS OF THE ARGUMENTS MUST BE EXECUTED TO OBTAIN BOTH HALVES OF THE DOUBLE LENGTH PRODUCT IN 58 μ SEC
^d PRODUCT ENCODED DATA, $a = 15$ GIVES THE EQUIVALENT OF FOUR REDUNDANT BITS

WEIGHTED DESIGN CONSIDERATIONS FOR AEROSPACE COMPUTERS

Three design criteria seem to deserve heaviest weighting because of their inseparable interrelationship in the design process. They are reliability, producibility, and number of LSI types. Of secondary importance are LSI package size, number of terminals per LSI type, testability and operational environment. Considerations which seem to be beyond the scope of present effort are special circuit design, system integration and the detailed packaging method.

Reliability

The simultaneous requirements for high reliability for short missions as well as long mean-life are difficult to satisfy with the same system. The best methods for realizing high reliability for a short mission use Massive schemes such as triple modular redundancy (TMR). This scheme is used in the IBM Saturn IB/V launch vehicle digital computer (LVDC) and launch vehicle data adapter (LVDA) where the requirement is for reliability of 0.996 for 250 hours (Ref. 5). Massive redundancy implies that the redundant components operate continuously to mask faults so that a critical function cannot be lost during a crucial part of the mission. A disadvantage of the triplicated Massively redundant system is that the replicated parts consume power up to at least three times that of a simplex system.

On the other hand, a switched redundant system such as the JPL-STAR computer has a potential for long mean-life by carrying along spares which are switched into the system upon detection of a failure. A fact which enhances system mean-life for the replacement type system is that semiconductor power-off failure rates are significantly lower than for power-on.

The NASA-referenced modular computer breadboard (MCB) represents an attempt to embody both principles and thus gain the advantages of each. It is intended that the arithmetic processors be capable of triplicated operation for boost phases and capable of reconfiguration for mission phases during which high reliability is not required. A chief advantage of this system which may be classed as a replacement type self-repairing system is the reduction in power afforded by switching off unused portions of the system. This is ultimately reflected in weight minimization for the on-board power source.

An important parameter of the stand-by replacement system is the relative failure rates for stand-by and operating parts. In order to examine the effects of the stand-by failure rate, a reliability model has been adapted from Kletsky (Ref. 6).

Kletsky's system is made up of N identical modules, L of which must be operating. If the operating failure rate of a module is λ_{μ} , and the ratio of power-off to power-on failure rate is α , then the system mean-life is

$$M = \sum_{K=0}^{N-L} \frac{1}{(L - K\alpha) \lambda_{\mu}}$$

By modifying the model to more closely resemble the system at hand, the effects of stand-by failure rate on system mean-life can be examined. The model modification is indicated in Table VII. The system is broken into separate but identical blocks, some of which are in stand-by. As shown in the first row of the table, the mean-life of a system with one stand-by element varies between 1.5 times and 1.91 times the mean-life of the un-replicated system as the failure rate ratio for power-off to power-on decreases. It is clear that the optimum partitioning of the system is strongly dependent on the ratio α . Information about stand-by failure rates for aerospace computers is available from two sources.

A 1965 report by Nerber (Ref. 7) of IBM indicates that there is a significant ratio of power-off to power-on failure rate. Nerber analyzed field failure data for over 100 guidance computers. These were undoubtedly transistor rather than integrated-circuit computers.

Unfortunately, it is not possible to compute α directly from the data given by Nerber. He gives only the ratios of the duty cycles he observed and the ratio of conventional MTBF's. We really need the explicit duty cycles rather than their ratios. On the other hand, he does give a clue to the actual duty cycles; "The observed duty cycles are in the lower third of the total range." Using this clue, we can compute a value for the ratio of power-off to power-on failure rate ratio, α .

Examining his data, we find that for systems of equal "vintage", the conventional MTBF for a second group of computers was 2.7 times that of a first group while the duty cycle ratio of group 2 to group 1 was 4.0. If a maximum duty cycle of 33% is assumed for group 2, the maximum value of α which he must have found is computed to be 0.33

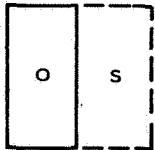
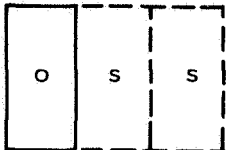
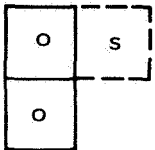
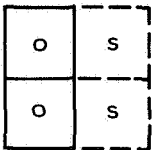
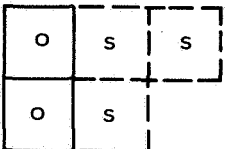
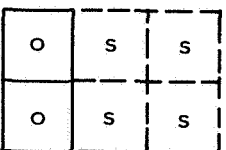
A more recent analysis of Minuteman II computer failures by Watson (Ref. 8) shows that the expected value of power-off to power-on failure rate ratio for integrated circuits is 0.55. A lower bound for this ratio appears to be 0.114. Extrapolation indicates that the ratio will decrease as more data is gathered. At any rate, present evidence from both sources indicates that the proper ratio for conservative integrated-circuit computer design lies between 0.1 and 0.5.

What appears to be needed for the design of computers with both high reliability and long mean-life is a methodology. The following is offered. The method is separated into steps a through g. Steps a and g require a good deal of intuition while the other steps are straightforward.

- a) Partition system into "natural" subsystems,
- b) Compute reliability required for each subsystem.

TABLE VI. MEAN LIFE COMPARISON

81119-39

<div style="display: inline-block; transform: rotate(-45deg); transform-origin: center;"> <div style="display: inline-block; transform: rotate(45deg);">FAILURE RATE RATIO</div> <div style="display: inline-block; transform: rotate(-45deg);">SYSTEM CONFIGURATION</div> </div>	$\alpha = 1.0$	$\alpha = 0.5$	$\alpha = 0.1$
	$M = \frac{1.5}{\lambda}$	$M = \frac{1.67}{\lambda}$	$M = \frac{1.91}{\lambda}$
	$M = \frac{1.835}{\lambda}$	$M = \frac{2.17}{\lambda}$	$M = \frac{2.82}{\lambda}$
	$M = \frac{1.66}{\lambda}$	$M = \frac{1.80}{\lambda}$	$M = \frac{1.96}{\lambda}$
	$M = \frac{2.16}{\lambda}$	$M = \frac{2.46}{\lambda}$	$M = \frac{2.86}{\lambda}$
	$M = \frac{2.56}{\lambda}$	$M = \frac{3.04}{\lambda}$	$M = \frac{3.74}{\lambda}$
	$M = \frac{2.90}{\lambda}$	$M = \frac{3.54}{\lambda}$	$M = \frac{4.58}{\lambda}$

λ = FAILURE RATE OF OPERATING PORTION OF SYSTEM
 α = RATIO OF POWER-OFF TO POWER-ON FAILURE RATE
 M = SYSTEM MEAN-LIFE

- c) Decide which, (if any) subsystems require redundancy to achieve this reliability.
- d) Decide whether subsystem reliability should be achieved by switching or Massive redundancy.
- e) Apply redundancy.
- f) Compute mean-life.
- g) Iterate design as required.

A simple example of the application of the system design methodology is given below. The system is composed of two "natural" subsystems which might for instance be a processor with failure rate $\lambda_1 = 0.00025$ failures per hour and a control unit with failure rate $\lambda_2 = 0.00004$ failures per hour. For the sake of this illustration, the two will be called subsystems A and B respectively. Assume that the following requirements exist:

- 1) System Reliability = 0.98 for a 250 hour mission
- 2) System Mean-Life = 10,000 hours.

First, the required reliability for each of the subsystems is computed by

$$R = \sqrt[n]{\text{system reliability}}$$

For two subsystems:

$$R_A = R_B = \sqrt[2]{0.98} \approx 0.99$$

Subsystem B has reliability

$$\begin{aligned} R_B &= e^{-\lambda_2 t} \\ &= e^{-(0.00004)(250)} \\ R_B &\approx 0.99 \end{aligned}$$

which is sufficient without redundancy. Subsystem A requires redundancy to achieve

$$R_B \geq 0.99$$

If Massive redundancy is used, a two-out-of-three system is characterized by

$$\begin{aligned} R_{A_1} &= 3e^{-2\lambda_1 t} - 2e^{-3\lambda_1 t} \\ &= 3e^{-0.125} - 2e^{-0.1875} \\ &\approx 0.99 \end{aligned}$$

The mean-life of the system is then computed by

$$\begin{aligned}
 M &= \int_0^{\infty} [R_{A1}(t)] [R_B(t)] dt \\
 &= \int_0^{\infty} \left\{ \left[3e^{-2\lambda_1 t} - 2e^{-3\lambda_1 t} \right] e^{-\lambda_2 t} \right\} dt \\
 &= \frac{3}{2\lambda_1 + \lambda_2} - \frac{2}{3\lambda_1 + \lambda_2} \\
 &\approx 3,040 \text{ hours}
 \end{aligned}$$

The mean-life falls short of the required 10,000 hours. Obviously, the problem lies with subsystem A. Actually, the application of Massive redundancy reduced system mean-life below what it would have if no redundancy had been applied. The mean-life for a system with no redundancy would be 3,050 hours.

If switching redundancy is applied to subsystem A with L replicas operating and N-L in stand-by, the reliability according to Kletsky (Ref 6) is:

$$R_{A2} = 1 - L^{-1} \left[\frac{1}{S} \prod_{k=0}^{N-L} \frac{(L + k\alpha)\lambda_1}{S + (L + k\alpha)\lambda_1} \right]$$

which can be reduced to:

$$R_{A2}(t) = \sum_{k=0}^{N-L} e^{-(L + k\alpha)\lambda_1 t} \prod_{j=0}^{N-L} \frac{\frac{L}{\alpha} + j}{j - k}, \quad j \neq k.$$

For the case in point,

$$N = 3$$

$$L = 1$$

$$\alpha = 0.1 \text{ (See references 4 and 6)}$$

$$\lambda_1 = 0.00025$$

Then

$$R_{A2}(t) = 66e^{-\lambda_1 t} - 120e^{-1.1\lambda_1 t} + 55e^{-1.2\lambda_1 t}$$

Multiplying and integrating to get system mean-life

$$\begin{aligned}
M_{sys} &= \int_0^{\infty} [R_{A2}(t)] [R_B(t)] dt \\
&= \frac{66}{\lambda_1 + \lambda_2} \frac{120}{1.1 (\lambda_1 + \lambda_2)} \frac{55}{1.2 (\lambda_1 + \lambda_2)} \\
&\approx 9460 \text{ hours}
\end{aligned}$$

Which is close enough to the required 10,000 hour mean-life.

The reliability for subsystem A with switching redundancy is:

$$R_{A2}(t) \approx 0.9919$$

This is roughly equal to the $R_{A1}(t)$ with Massive redundancy.

Producibility

Maurer and Ricci (Ref. 9) have recommended 100 gate TTL arrays with fixed interconnection for on-board guidance computers for the 1970-72 period. This represents a two to two-and-one-half derating of the expected industry capability for that period. Petritz (Ref. 10) has indicated that fixed wiring arrays of 250 gates will be available. The first attempt at partitioning was made using a maximum of 250 - 300 gates per array. For this size of array, considerable interconnecting of circuit elements on the array is possible. During the second part of this study, about 100 gates per array will be the partitioning goal. The gate-to-lead-bond ratio is expected to deteriorate to the extent that a 100-gate array will require about 80 percent as many lead-bonds as a 250-gate array.

Number of LSI Types

The number of LSI types is also dependent upon the "natural" partitioning of the machine. Ways to minimize the number of types always require that the array have gates which are useless for some applications in order that the array be useful for many applications. This consideration must be exercised on the first pass design which produces the "natural" functions to which the reliability methodology is applied.

LSI Package Size

Package size is determined by the size of the chip and the number of pins. One 156 pin square prototype package has an area of approximately 2.5 in². A 50-pin package whose hermeticity is considerably better is 1" x 2.5". Square packages with pins on all four sides tend to make heat management more difficult, but it is assumed that the packaging problem per se will be studied more fully in continuations of the present study.

Number of Terminals per LSI Type

Hermetically sealed packages are now available in up to 50-pin versions. Also available are 128 and 156-pin plastic packages. While hermetic seals for the larger packages will be more difficult to maintain, it is expected that hermeticity of these packages will be sufficient.

Testability

The only design criterion for the logic arrays is that each memory element in the array be capable of being set to a known state. This avoids many of the unsolved diagnosis problems. The generation of tests for a defective array should be part of the design process, but can be delayed to fairly late in the design iteration cycle.

Operational Environment

Aerospace applications have required cooling via cold plate or cooling air. Generally, the worst environments with respect to temperature and humidity are aircraft. Ordinary cooling methods are able to keep array temperatures at safe levels with as much as 2.5 watts per 250 gate array with cold plate cooling (Ref. 11).

Special Circuits

Integration of special circuits is considered highly important. First priority should be given to memory circuits with either plated wire or coincident-current core memory.

System Integration

System integration considerations, while of great importance, will receive attention only as they relate to the other design considerations. It is expected that system integration per se will receive greater attention in subsequent phases of the study.

Packaging Method

Large cavity flat packs are assumed. Packaging studies will also be a secondary consideration in this phase of the study.

REFERENCES

- (1) Hamilton Standard Report HSER5081, Seventh Quarterly Progress Report for Research on Advanced Kick Stage Guidance Computer Study, 1 February 1968 to 30 April 1968.
- (2) United Aircraft Drawing No. 24000, "MCB Block Diagram", approved 8 March 1968.
- (3) Baechler, D. O., "State of the Art of Aerospace Digital Computers, 1962-1967", Computer Group News, Vol. 2, No. 1, January 1968.
- (4) Hooper, R. L. and Amdahl, L. D., "Trends in Aerospace Computers", Datamation, November, 1967.
- (5) Anderson, E. J. and F. J. Macri, "Multiple Redundancy Applications in a Computer", Proceedings of the 1967 Annual Symposium on Reliability, January, 1967, pp. 553-562.
- (6) Kletsky, E. J., "Upper Bounds on Mean Life of Self-Repairing Systems", IRE Transactions on Reliability and Quality Control, October 1962, pp. 43-48.
- (7) Nerber, P. O., "'Power-off' Time Impact on Reliability Estimates", IEEE International Convention Record, Part 10, March 22-26, 1965, New York, N.Y., 1965, pp. 1-8.
- (8) Davis, L. K., G. A. Watson, and T. G. Schairer, "Advanced Computer Dormant Reliability Study, Final Report", Autonetics Division of North American Rockwell Corp., October 14, 1967.
- (9) Maurer, H. E., and R. C. Ricci, "Horizons in Guidance Computer Component Technology", IEEE Transactions on Computers, July 1968.
- (10) Petritz, R. L., "Current Status of Large Scale Integration Technology", 1967 Fall Joint Computer Conference, AFIPS Proceedings, pp. 65-85.
- (11) Jennings, R. C., "Design and Fabrication of a General Purpose Airborne Computer Using LSI Arrays", Digest 1968 Computer Group Conference, IEEE, pp. 50-53.

APPENDICES

APPENDIX A

IBM 4 P1-EP INSTRUCTION LIST

<u>Fixed Point Instructions</u>	<u>Execution Time (usec.)</u>
Load	1.9
Load	5.0
Load Halfword	5.0
Load and Test	1.9
Load Complement	2.1
Load Positive	2.1
Load Negative	2.1
Add	2.1
Add	5.0
Add Halfword	5.4
Add Logical	2.1
Add Logical	5.0
Subtract	2.1
Subtract	5.0
Subtract Halfword	5.4
Subtract Logical	2.1
Subtract Logical	5.0
Compare	2.1
Compare	5.0
Compare Halfword	6.0
Multiply	9.2
Multiply	10.4
Multiply Halfword	11.6
Divide	20.0
Divide	20.8
Store	5.0
Store Halfword	5.0
Shift Left Single	Variable
Shift Right Single	Variable
Shift Left Double	Variable
Shift Right Double	Variable
<u>Logical Instructions</u>	
Compare Logical	1.9
Compare Logical	5.0
Compare Logical	5.4
And	3.4
And	5.0
Or	3.4
Or	5.0
Exclusive Or	3.4
Exclusive Or	5.0
Test Parity	5.0

APPENDIX A
(continued)

<u>Logical Instructions</u>	<u>Execution Time (usec.)</u>
Test Under Mask	5.0
Sumcheck	Variable
Insert Character	5.0
Store Character	5.0
Load Address	2.9
Shift Left Single Logical	Variable
Shift Right Single Logical	Variable
Shift Left Double Logical	Variable
Shift Right Double Logical	Variable
 <u>Branching Instructions</u>	
Branch on Condition	4.2
Branch on Condition	4.4
Branch and Link	4.0
Branch and Link	4.1
Branch on Count	4.2
Branch on Count	4.4
 <u>Status Switching Instructions</u>	
Load Program Status Word (PSW)	7.5
Load PSW Special	9.0
Set Program Mask	2.1
Set System Mask	5.0
Change Priority Mask	5.4
Supervisor Call	15.0
Set Storage Key	4.5
Insert Storage Key	5.0
 <u>Input/Output (I/O) Instructions</u>	
Start I/O	Variable
Test I/O	Variable
Halt I/O	Variable
Test Channel	Variable
Read Direct	Variable
Write Direct	Variable

APPENDIX B

HONEYWELL ALERT INSTRUCTION LIST

<u>MNEMONIC/ OPERAND</u>	<u>INTERPRETATION</u>	<u>EXEC. TIME (usec)</u>
ADD X, V	Add to Accumulator	2
ADM X, V	Add to Memory	7
AIX X, V	Augment Index Immediate	3
AUX X, V	Augment Index	3
BAR X, V	Branch and Return	6
CSK X, V	Character Skip if Equal	5
DIV X, V	Divide	30
DJX X, V	Decrement and Jump on Index Not Zero	2
DID X, V	Double-precision Load	3
DST X, V	Double-precision Store	10
EXC X, V	Execute	1
EXT X, V	Extract	2
HAD X, V	Half ADD	2
HLT	Halt	2+Δ
JAN X, V	Jump on Accumulator Negative	2
JAP X, V	Jump on Accumulator Positive	2
JAZ X, V	Jump on Accumulator Zero	2
JIX X, V	Jump on Index not zero	2
JMP X, V	Jump unconditionally	2
ICH X, V	Load Character	2
IDA X, V	Load Accumulator	2
IDB X, V	Load B-register	2

APPENDIX B
(continued)

<u>MNEMONIC/ OPERAND</u>	<u>INTERPRETATION</u>	<u>EXEC. TIME (usec)</u>
LDX X, V	Load Index Register	2
LIM X, V	Load Interrupt Mask	2
MPY X, V	Multiply	12
PAS	Pass	2
CMP	Complement contents of the accumulator	2
ABS	Set contents of accumulator positive	2
XAB or XBA	Exchange contents of A and B register	2
TWA	Transfer (RWC current address) to A.	2
TBA	Transfer contents of B-register to the accumulator	2
TAB	Transfer contents of accumulator to the B-register	2
TAX V	Transfer (A) to XR_V	2
TXA V	Transfer (XR_V) to the accumulator	2
AIC N	Shift accumulator left cyclic	Variable
ARC N	Shift accumulator right cyclic	Variable
ALS N	Shift accumulator left arithmetic	Variable
ARS N	Shift accumulator right arithmetic	Variable
LLC N	Shift AB left cyclic	Variable
IRC N	Shift AB right cyclic	Variable
LLS N	Shift AB left arithmetic	Variable

APPENDIX B
(continued)

<u>MNEMONIC/ OPERAND</u>	<u>INTERPRETATION</u>	<u>EXEC. TIME (usec)</u>
LIP N	Shift AB left arithmetic and protect sign of B	Variable
IRS N	Shift AB right arithmetic	Variable
IRP N	Shift AB right arithmetic and protect sign of B	Variable
SRB X	Set/Reset Interrupt Block	2
SIB	Set Interrupt Block	2
RIB	Reset Interrupt Block	2
SKN X	Skip if signal is not set	3
SKC X	Control and Skip	4
STE X	Set External Signal	2
SKE X	Skip if External Signal is not set	3
PIN X, V	Peripheral input and skip	8 if skip 4 if no skip
POT X, V	Peripheral output and skip	4
PDT X, V	Peripheral Data Transfer	Variable
PDT X, V	Peripheral Data Transfer	Variable
PCB X, V	Peripheral Control	Variable
RSS X	Restore Status	2
SCH X, V	Store Character	7
SKM X, V	Skip if Accumulator and Memory are equal	4
SKX X, V	Skip on Index High	4
SMP X, V	Superimpose (Inclusive OR)	2

APPENDIX B
(continued)

<u>MNEMONIC/ OPERAND</u>	<u>INTERPRETATION</u>	<u>EXEC. TIME (usec)</u>
SMZ X, V	Skip if memory is zero	3
SST X, V	Substitute	7
STA X, V	Store Accumulator	6
STB X, V	Store B-register	6
STI X, V	Store Interrupt Register	6
STS X, V	Store Status	7
STX X, V	Store Index register	6
SUB X, V	Subtract from Accumulator	2
SXI X, V	Skip immediate on Index high	3
TIY X, V	Tally	7
XML X, V	Exchange Interrupt Mask	6

APPENDIX C
A REVIEW OF LSI TECHNOLOGY
FOR AEROSPACE COMPUTERS

The advancing capability of the semiconductor industry to fabricate and interconnect 100 or more logic gates on a single silicon chip (i.e., large-scale integration) will have a significant effect on the organization and design of future digital systems. New requirements on gate-to-pin ratio, standardization, discretionary interconnect, testability, and redundancy applications must be met if systems are to use this advanced technology with efficiency and economy. The interconnection of at least 100 gates on a single semiconductor chip (or full slice) with a gate-to-pin ratio of at least two can be considered a basic definition of large scale integration (LSI). This technology offers the following advantages:

- | | |
|--------------------|--|
| 1) Cost | 1970, 2.5¢ per gate in 100 gate arrays
1974, 1.0¢ per gate in 100 gate arrays
(Noyce, Reference 1) |
| 2) Size | 1000 good gates per 1.5-inch wafer |
| 3) Reliability | More batch processing, less external interconnect |
| 4) Standardization | Entire subsystems on single wafer |
| 5) Power | Less drive required on wafer |
| 6) Speed | Shorter line lengths, less capacitance |
| 7) Packaging | Less interpackage wiring |

In order to obtain the above advantages, however, certain requirements on system organization must be met.

LSI System Implementation Requirements

In order to realize the increased economy and reliability of systems implemented in large chip or full wafer LSI, six requirements must be satisfied:

- 1) Systems must be organized and partitioned to obtain a high gate-to-logic-pin ratio in order to maximize the use of wafer components.
- 2) Efficient use must be made of standard logic cells more complex than current integrated circuit chips to obtain the economy of batch fabricated processing and interconnect.

- 3) Logic cells must be defined to both facilitate automated routing and to allow automated testing with a restricted number of test points.
- 4) Discretionary interconnection of logic elements must be eliminated or minimized.
- 5) Sufficient redundancy must be used to ensure reliability, standardize logic cells, facilitate automated testing, and allow economical interconnect in view of non-100 percent yields.
- 6) Logic, including data flow paths and control, must be regularized and standardized to obtain the economy promised by mass production of LSI.

The following paragraphs describe these requirements in more detail and discuss certain approaches to their solution.

The requirement for a high gate-to-pin ratio exists because the LSI component processing capability in a pin-limited package is beyond what can be utilized by most system designs. That is, LSI wafers of 2000 usable gates may be packaged with 150 external pins which suggests a desired gate-to-pin ratio of 10-to-15. But the typical ratio for current systems is only 0.8-to-5.0. Therefore, system organization must obtain a more efficient logic partitioning to meet the LSI wafer capabilities. In this respect, redundancy can be effectively used in the system partitioning to increase the utilization of available components. Examples of system organizations aimed at increasing the logic gate-to-pin ratio include: 1) RCA's LIMAC "distributed control", 2) Litton's "block-oriented computer" with serially addressed arithmetic units on a common bus, and 3) Hughes' minimization of control logic by modular asynchronous processing units.

The use in future computer systems of more complex logic cells than current integrated circuit chips will allow proportionately less discretionary interconnect between good cells. The limitation on cell complexity is primarily the much-reduced yield of higher complexity cells. Other tradeoffs in the definition of cell size are more components per area and simpler system tests for large cells against more components per wafer and higher logic efficiency for smaller cells.

Since an LSI wafer will probably be pin-limited, built-in test circuits providing fault detection signals can increase reliability while facilitating the testing of wafers with limited access. The ability to easily test a logic cell of an LSI wafer must be strongly considered in the definition and design of the cell.

Discretionary interconnection of logic elements must be eliminated or minimized. Current approaches to this problem follow:

1) Discretionary Signal Routing and Mask Making (Texas Instruments)

- . Use of the multiple level interconnect generator (the MIG system) allows semiautomated wafer routing.
- . At least one hour of IBM 7044 computer time is required per wafer.
- . Six unique masks are required for each wafer.
- . Wafer failures result in loss of the computer routing and mask costs expended.
- . Unique problems of line lengths, feedthroughs, mask alignment, etc., occur.

2) 100 Percent Yield (RCA, Fairchild, Signetics, et. al.)

- . Only limited size cells of 100 percent yield can be obtained. A large sample of a leading vendor's high yield quad two-input NAND gate chips, for example, showed that 0.5 percent of the 100-gate cells and none of the 144-gate cells had 100 percent cell yields.
- . Although lines within a cell can be routed by standard processing techniques, the 100 percent yield cells must be separated from the wafer or interconnected by discretionary techniques.
- . The majority of good components are not used because they do not occur in cells of 100 percent yield.
- . The approach is really a medium-scale integration technique to increase the complexity of system components beyond present integrated circuit chips.
- . As a true LSI approach, it is both costly and inefficient of components.

3) Block-Oriented Approach (Litton)

- . No discretionary signal routing is required.
- . A simple discretionary address key is used external to the wafer to selectively address good cells. Additional wafer pins are required for the addressing.
- . Power is separately and selectively applied to the good cells which increases the number of required wafer pins.

- A significant loss (factor of 5 or greater) in computing speed results from the fact that common control and data busses must be used in a sequential manner to access the cells.
- Only a limited set of programs could be partitioned into the independent calculations which are necessary to obtain system efficiency from the block-oriented approach.

Development of IC and LSI Technology

To place this discussion in the proper perspective, it is best to review past history in the technology areas affecting aerospace systems. Too often, technology projections dwell on possible rather than on probable developments. Two factors, other than purely technical elements, must be kept in mind in projecting technological advances. These are the probable speed of technological development and the force of this development.

Consider the speed of development of present integrated circuits (IC). The first efforts on integrated circuits began in early 1957. These circuits have just recently reached military operational status.

The following three basic factors contributed most to the rapid application of IC: early military support of IC research and development programs, early decision to use IC in the Minuteman missile, and early improved performance at equivalent price of nonmilitary digital circuits. This development of IC from laboratory through the first system followed a typical path. However, the real impetus that removed IC from high priced, specialized military parts so early was the commercial and industrial, not military, potential. This was probably not the case with transistors, but will be more the case with large scale integration (LSI). The commercial potentials will influence the general industry developments to a much higher degree early in the life of major technological advances.

The reason for this is the changed nature of military procurement. The emphasis on fixed-price contracts has forced a conservatism upon military electronics suppliers that may result in lower immediate costs, but also tends to delay the moving of major technologies from the laboratory to the field. At the same time, private investments are tending toward the commercial industrial areas.

Three recent examples of this trend may be found in IC packaging advances and dielectric isolation techniques. The two major packaging innovations in recent years in IC's have been the dual in-line package (DIP) and the plastic package. Both were motivated by commercial market needs. No equivalent advances have been made, for example, in very high power, small size, high-reliability packaging for military application. Until recently, dielectric isolation has been studied by the major IC companies primarily for application to extremely high-speed, commercial digital computers, and to a

much lesser degree for radiation hardening. The little radiation hardening effort done has been only at the level directly funded by the military in small research and development contracts.

The following considerations should be borne in mind when projecting advanced technology developments: new technology will reach maturity based primarily upon nonmilitary factors, technology related to cost will receive more emphasis than technology related to performance, and technology related to high volume will receive more emphasis than that related to specialized markets.

LSI cost projection in the area of 1¢ per gate are common for the 1970 to 1980 era. Quotes of 15¢ per gate were received two years ago for a full military specification part on a standard IC. If the initial development and nonrecurring tooling costs are ignored, and if very large volume production is assumed, then the 1¢ per gate is possible. Without specific effort on the unique aerospace problems, such potential may not be realized in time to benefit 1970 to 1980 systems.

What is LSI? One of the first considerations is whether present IC or LSI must be assumed. The distinction between these two areas and a third, hybrid microelectronics, is at best vague and somewhat confused due to promotional statements made by many company marketing organizations. A clarification of the distinctions between these terms will help place this discussion in the proper perspective.

Integrated circuits, or more exactly monolithic integrated circuits, differ from single transistors in that a circuit function is fabricated by batch processing methods with a very limited number of materials (silicon, silicon dioxide, aluminum), and processes (diffusion, bonding, and photolithographic techniques). The successful application of these techniques, while imposing circuit limitations (no inductors, few capacitors, no high value resistors), has permitted new circuit design freedoms (ability to use many transistors and tailor each transistor specifically for one circuit only). It has resulted in overall economic and reliability gains over discrete component designs. The level of complexity of a given IC is set primarily by economic (yield) considerations. As the industry improves its yield, the level of complexity, as set by economic considerations, increases. Thus, it would appear artificial to set the definition of an IC by measuring its complexity. Also, if in the batch processing the materials and process limitations are violated, the end product can probably not be classified as an IC.

By this reasoning, an IC or LSI is not measurable in terms of complexity alone. The terms, "hybrid IC" and "hybrid LSI" are contradictions because hybrid circuits are merely collections of many materials and processes.

If LSI denotes a new item, there must be something more required to distinguish it from any IC or hybrid circuit. There are at least two new factors that permit this distinction. One is the ability to fabricate a part even though the part does not contain all operating subparts; this is the discretionary wiring approach. The other is the system design requirement. If an LSI part is a small system or subsystem, then its use is probably in some way tailored for that specific application.

On the basis of these distinctions, if a shift register, however large, is economical enough to produce on a 100 percent yield basis, it is not an LSI part. Also, a collection of IC chips all mounted in one package, however complex, is not an LSI part, but merely a small IC "black box".

The implication is that a careful system design is required--a design that cannot only exploit the potential of advanced microelectronic technology, but do so in a manner consistent with good system design practices, while retaining advantages of cost and reliability within the limitation of requiring only a few common parts. The requirement for low quantity with no increase in the number of unique parts is the problem that must be solved before LSI can be successfully used. It is also the requirement that distinguishes LSI from complex IC.

Device Considerations

Potential of MOSFET Versus Bipolar Devices--If LSI is used, the next question is "what devices will solve the majority of the system requirements?" Bipolar devices used in the majority of today's microelectronic systems, have two disadvantages for meeting future requirements. They are high power devices and are potentially unreliable.

The high power limitation cannot be removed without several major simultaneous process breakthroughs. The technology to build very large resistors on a wafer, methods to reduce overall device size including isolation areas, and improvement in low current gain characteristics are required.

Since Bipolar devices are current-controlled, the chances for eliminating one particular basic failure mechanism are few. This failure mechanism is the finite lifetime of the aluminum metalization due to film temperature and current density. (Reference 2)

In contrast, metal-oxide-silicon field effect transistors (MOSFET) are voltage-controlled, are capable of being made much smaller than bipolar devices, and require no isolation regions. In addition, many fewer process steps are required aiding in obtaining comparable yields for much higher complexity.

In considering low power applications for MOS circuitry, there are several tradeoffs between p-MOS, p-MOS multiphase, n-MOS, ion-implanted MOS, complementary, and complementary on sapphire types. The complementary type, by virtue of its inherent zero, or negligible power drain provides the lowest power circuitry for general application. However, it presents problems that are still to be solved in fabrication. In addition, its use results in power loss due to switching transients driving capacitance loads, thus affecting higher speed applications. The capacitance load can be reduced by fabricating complementary MOS on sapphire. Additional processing problems are envisioned, however, which would probably result in the first circuits not being available in moderate quantity and medium complexity for about one year.

Multiphase p-MOS in shift register applications operate at as low power as complementary MOS, but is useful only in shift register applications as a power saving effect. P-MOS multiphase shift registers are available now and work well and reliably.

P-MOS logic can be operated at low frequency as a means of reducing power and is presently available. It will always have substantially higher power than complementary MOS operating at the same speed. While similar processing steps are required for both p-MOS and n-MOS, p-MOS has received more emphasis because it is somewhat easier to manufacture.

For the 1970 to 1980 era, the complementary silicon on sapphire would appear to offer the most promise. Depending upon system requirements, complementary device parts can be made, at a slight increase in processing complexity, permitting even less power dissipation than single-device type MOSFET circuits while permitting increased circuit speed. MOSFET devices with complementary pairs used where appropriate would appear to be indicated for future development. Companies such as RCA, Westinghouse, and Hughes are now producing prototype complementary MOS circuits. Experimental ion-implanted MOS devices are also available. They offer at least a three-fold increase in speed over conventional MOS.

Device Packaging and Application Requirements -- The advancing technology influencing computer development for the ensuing ten years is underscored by the influence of batch fabrication. With the objectives of cost, reliability, performance, maintainability, and size improvements, the use of MSI by the early 1970's and full LSI in the late 1970's will be evident in operational aerospace computers. Current hybrid thick and thin film technology, for input/output circuitry, higher power circuits, and prototype circuitry will continue to play a relatively small part in the design and fabrication of production aerospace computers. Integrated circuits will provide the major innovation in production computers going to the field in the early 1970's (e.g., Hughes' HCM-205; Litton L-3050; and RCA VIC-36A, all presently in development.) New designs configured during the next few years will use LSI designs completed by 1973 to 1974, and those reaching the operational scene by the mid or late 1970's will use full LSI for arithmetic units, logic, control, and memory.

MOS technology offers lower power, potentially higher yields because of simpler processing, 4 to 10 times size improvement, and resultant cost savings stemming from processing, yield, and density gains. Of all usage (MOS and bipolar), MOS will find a 30-percent utilization in the central processor by 1973 and 50-percent by 1978; an 80-percent memory utilization of MOS over bipolar is seen by the late 1970's.

LSI entails the increase of logic per component as shown by Figure 41. This results from the advance in wafer technology that will bring by 1975 an increase in yield (see Figure 42) on the wafer, and a reduction in device size resulting from a combination of optical, mask making, and etching or diffusing technology. The increase in logic per wafer necessitates a partitioning and logic organization within the new constraint of pin limitation for the wafer. Since the cost of added LSI gates is inexpensive, the approach of interconnecting an entire function on the wafer, repeating some functions, and the use of new techniques for minimizing pins will be required. Figure 43 shows the projected gate-to-pin ratio trend through the 1970's for operational computers. The increase in gates per wafer brings a cost reduction attributable to batch fabrication at initial device diffusion as well as at subsequent logical interconnection. Projected cost, the most prominent inducement to this technological change, is plotted in Figure 44. Current projections indicate a cost advantage in favor of MOS circuitry over bipolar. The basic reason for this is the greatly increased complexity per unit area of silicon; a MOS transistor requires 10 to 100 mil² of silicon versus over 100 to 1000 mil² for bipolar. Also, only one-half as many process steps (or fewer) are required for the MOS device. The ultimate cost of MOS would seem to be bounded by wafer size and yield growth versus optical limits on individual device area and interconnection complexity; projections of these factors support continued reduction in unit cost for MOS as shown in Figure 44.

Reduced device size on the chip coupled with more devices on a single chip mean smaller signal transit times, shorter inter-gate connections, and reduced power. To capitalize on this speed improvement, minimum and optimized wafer interconnections must be used. Speed projections for the ensuing era of the 1970's are shown on Figure 45. For the near term, bipolar switching speeds are expected to remain in the 15 to 50 MHz range, but tend toward the low nanosecond range by the late 1970's. Although currently slower, complementary MOS devices using ion-implant techniques to eliminate the gate-to-drain overlap (thereby reducing Miller feedback capacitance) are expected to meet gate delays approaching three nanoseconds by the mid 1970's. In complementary MOS devices, almost all power is expended for transient operation; therefore, a linear gate delay-power constant can be plotted as shown in Figure 46.

Device density currently ranges from under 10 mil² for developmental units to 200 mil² for MOS devices, and 4 to 10 times greater than this for bipolar. MOS devices are interconnected within cells much the same as individual bipolar circuits. Typical cells range from 1600 to 5000 mil². The trend is expected to show an area reduction for MOS to under 10 mil²/gate in field equipment by the late 1970's. (See Figure 47.) Multilayer interconnection of the cells and circuits will form the desired function; new techniques, discretionary wiring, or 100 percent yield techniques will be used.

Bipolar integrated circuit failure rates have improved from approximately 0.4 percent per 1000 hours in 1962 to 0.003 percent per 1000 hours in 1967. The 1980 predicted reliability for MOS integrated circuits is 0.0001 percent per 1000 hours. Figure 48 illustrates the likely cross-over time for bipolar and MOS array failure rates to be between 1972 and 1973.

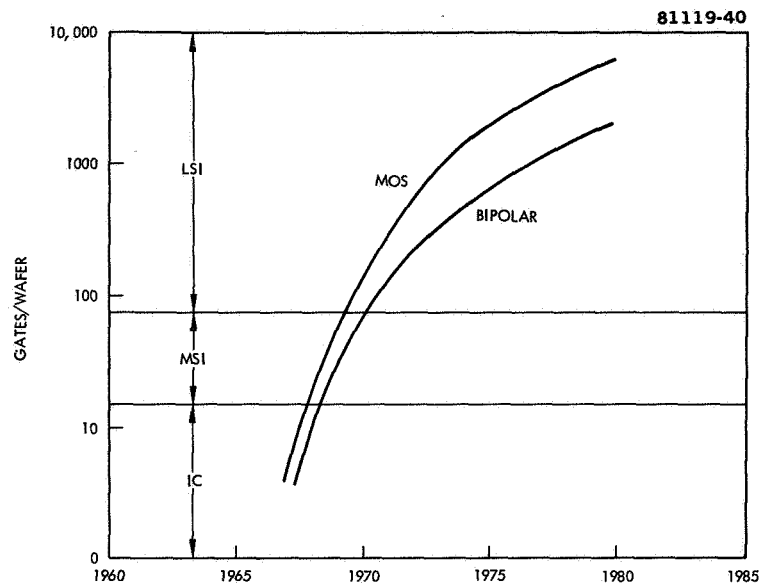


Figure 41. Logic per Wafer Computers

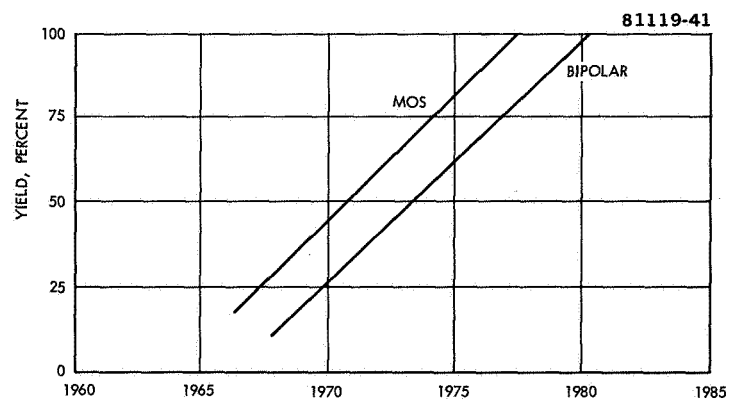


Figure 42. Device Yield on Individual Wafers

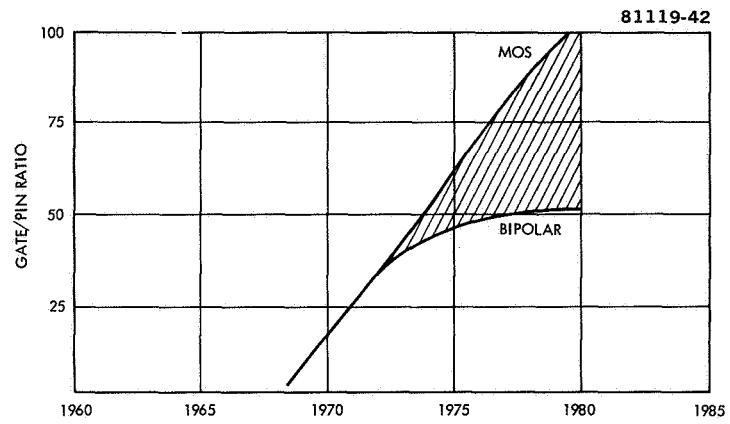


Figure 43. Gate/Pin Ratio for Components (Wafers)

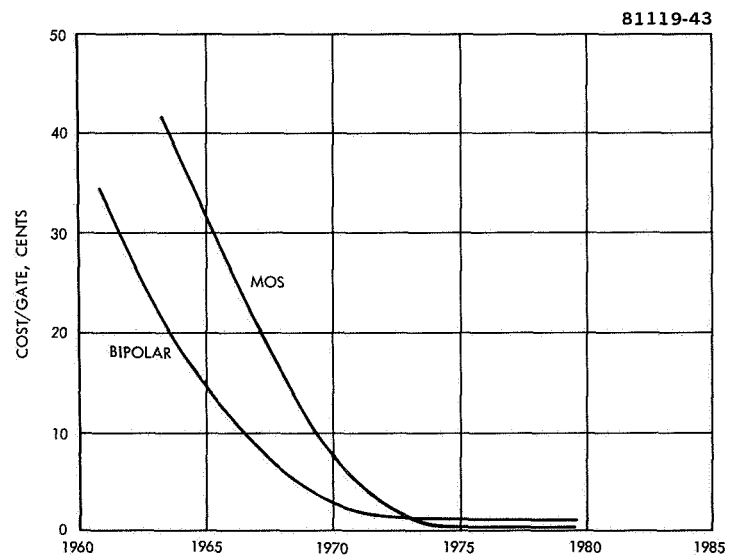


Figure 44. Cost/Gate — LSI in Computers

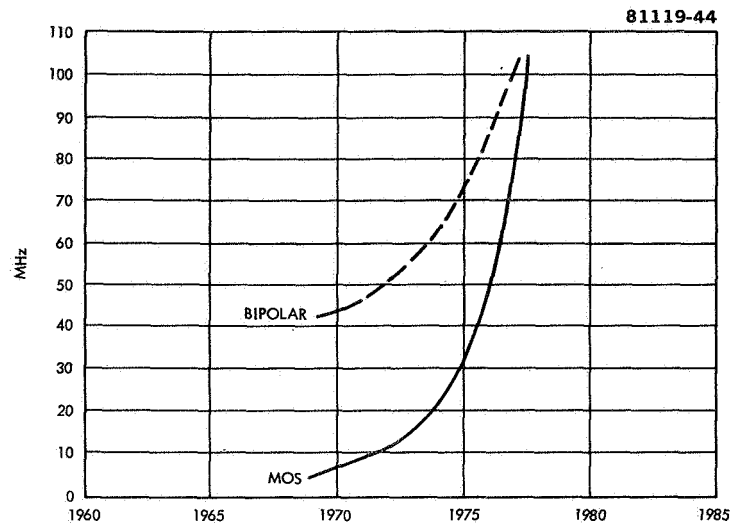


Figure 45. Logic Speed

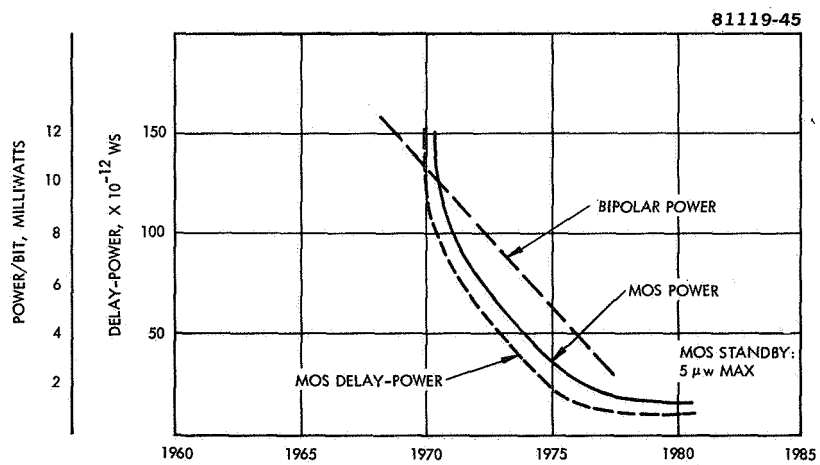


Figure 46. Delay-Power Figure-of-Merit/Bit (MOS), Power/Bit (Typical MOS and Bipolar)

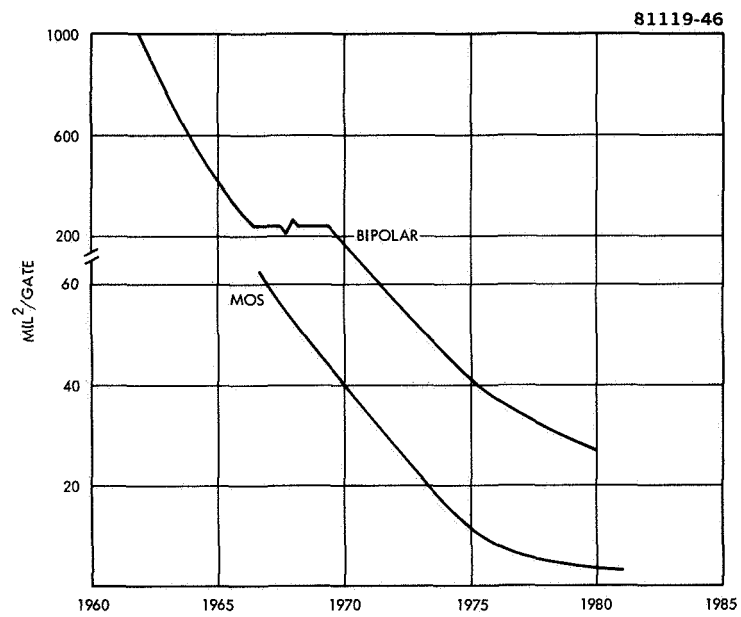


Figure 47. Area/Gate

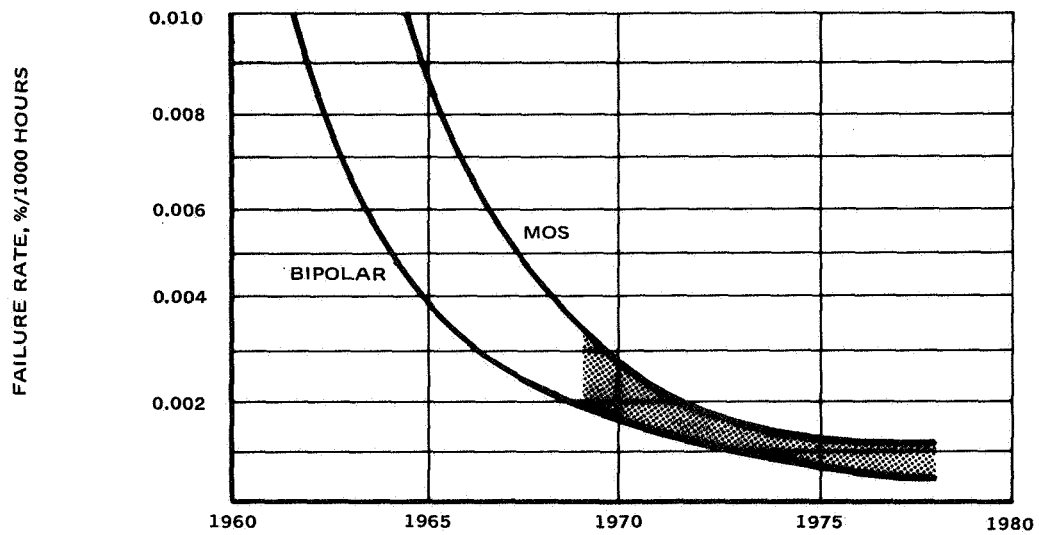


Figure 48. Device Failure Rate

Complexity of computer functions will increase by a factor of 2 to 4 by 1973 and 6 to 10 by 1980. This is attributable to the necessity to duplicate functions in an effort to retain signals within a given wafer. Also, added gates are available at little or negligible cost and power is an inducement to use them for greater performance (i.e., parallel operations). In addition, more self-test, fault isolation and repair circuitry will be incorporated.

Microelectronics (LSI) Problem Areas

The introduction of IC into computer design resulted in a drastic decrease in size and cost relative to discrete component machines. The trend toward more complex chips continues in the form of LSI toward the possible goal of a "computer on a chip". However, several problem areas have developed as a result of increasing complexity.

The primary problem is, of course, yield. As the chip size and complexity increases, the probability of a perfect or near perfect chip falls off very rapidly. Without considerable improvement in process technology, the "perfect chip" approach severely limits the maximum chip complexity. Several approaches for breaking down this complexity barrier have been proposed.

Techniques are being developed at several companies (including Hughes, Texas Instruments, and IBM) to develop a discretionary wiring system that will allow the randomly located bad circuits of a semiconductor wafer to be omitted from a two-layer interconnect, which forms a function block from the good circuits on a wafer. While each wafer has different internal interconnections, the wafers that have identical functions will appear identical. Theoretical studies show that random flaws in the basic silicon limit fixed-wiring schemes to chips of about 150 by 150 mils (depending on what constitutes an acceptable yield). With discretionary wiring, the silicon flaw is no longer the limitation. The yield on the interconnect and packaging is now the limitation. Present indications are that one-inch diameter slices with discretionary wiring will have sufficiently high yields to be feasible. The question of how to most economically achieve discretionary wiring remains to be resolved. The straight-forward use of present metallization techniques would require the generation of a new second metallization layout for each unique configuration of cell defects. This approach can be expedited by the use of computer-developed layouts with computer-controlled photoresist exposure. A proprietary plan being developed at Hughes greatly reduces the computer time required to develop the metallization configuration.

A study is presently being carried out at Litton on an entirely different approach. It is proposed that each chip should carry a multiplicity of arithmetic units usable as serial DDA's, or limited general processing units that form the components of a block-oriented computer. This approach improves the gate-to-pin ratio to the point where the selection of 'good' units can be accomplished by discretionary wiring external to the chip.

Another LSI problem area involves the interconnection pins. As chip complexity increases, the increasing number of external connections required becomes awkward. By carefully partitioning the circuitry so that each chip performs a more or less independent function, the interconnect density can be reduced. Computer-aided design can be used to advantage here.

As more gates are put in a package, it is noted that the number of leads per gate decreases. This results from the fact that some interconnections are entirely within the package. As a result of the reduced number of interconnections external to the IC packages, the computer reliability will increase, provided the more complex packages are not sufficiently less reliable. In choosing the level of integration that will be used for future aerospace systems, the complexity-cost and complexity-reliability tradeoffs must be carefully considered.

Increases in component density create the problems of greatly increased power density and greatly reduced access for maintenance. Power dissipation for state-of-the-art devices is typically 1 watt/in³; this could increase requiring very efficient heat transfer. Probably the only practical solution to the maintenance problem is the use of test circuits built in at the circuit and/or module level, since small size is making physical access to numerous test points increasingly difficult, and certainly impractical.

The success of microelectronics to date has been partially due to device/circuit standardization. To fully exploit the advantages of the emergent LSI technology, standardization at a higher functional level will be required. (Consider that a computer arithmetic and control unit may be built in 10 or even fewer packages within perhaps two years.)

Many projections as to the future of LSI contend that it will be limited to high production commercial machines for the near future because of the high design and tooling costs. The wide-spread entry of LSI into the military market could be hastened by the use of computer-aided design and computer-controlled mask generation to reduce initial costs and turnaround time. The automated design facility presently being organized at Hughes is expected to aid in this function.

References

- (1) Noyce, R. N., "Making Integrated Technology Work", IEEE Spectrum, May 1968, pp. 65.
- (2) Motorola, Inc., "Metalization Failures in Integrated Circuits", RADC-TR-67-477, September 1967.

APPENDIX D

NEW TECHNOLOGY

After diligent review of the work performed under this contract, no new innovation, discovery, improvement or invention beyond that mentioned in our proposal was made.

FR 69-11-621
92577